Universidad Carlos III de Madrid
Departamento de Informática

Tesis Doctoral

Predicción Local mediante Algoritmos Evolutivos

Autor: Cristóbal Luque del Arco-Calderón
Directores:
Pedro Isasi Viñuela, José Mª Valls Ferrán
Leganés, 2009

Universidad Carlos III de Madrid
Escuela Politécnica Superior

Predicción Local mediante Algoritmos Evolutivos

Tesis Doctoral

Cristóbal Luque del Arco-Calderón
Leganés, 2009

Departamento de Informática

Escuela Politécnica Superior
Universidad Carlos III de Madrid

Predicción Local mediante Algoritmos Evolutivos

AUTOR: Cristóbal Luque del Arco-Calderón

DIRECTORES:
Pedro Isasi Viñuela, José Mª Valls Ferrán
Leganés, 2009

Tribunal nombrado por el Mgfco. y Excmo. Sr. Rector de la Universidad Carlos III de Madrid, el día ..... de ..... de 2009.

Presidente: D. .....

Vocal: D. .....

Vocal: D. .....

Vocal: D. .....

Secretario: D. .....

Realizado el acto de defensa y lectura de la Tesis el día ..... de ..... de 2008 en .....

Calificación: .....

EL PRESIDENTE

LOS VOCALES

EL SECRETARIO

A Miguel Luque

Agradecimientos

Quiero expresar mi agradecimiento a todas las personas que me han ayudado y que han hecho posible que este trabajo haya podido concluirse, y en especial:

A mis directores de tesis, José María Valls y Pedro Isasi, por haber confiado en mí y mantener su fé, más allá de lo que yo incluso sería capaz de hacer.

A mis padres y hermano, por haberme animado y dado siempre todo lo que necesitaba. Por haber estado siempre ahí cuando me hacían falta.

A Lucía por apoyarme y estar conmigo desde el principio. Por no haber dudado jamás, y por haber permanecido fuerte.

A Yago, Juan David, David, César, Julio, Alejandro, Sandra, Inés, Ramón, Gustavo, Ricardo, Antonio, por haber sido compañeros y amigos durante todos estos años.

A Rafael, Saúl, Ramón Jesús, Juan Antonio, Luis, Rafa, Pepe, Gonzalo, Vanesa y Rocío, por su amistad.

A mi primo Chema, a Tenti, a mi padre, por iniciarme en el el misterio y la belleza de las matemáticas. A Francisco Castro, José Manuel Facenda, Ramón Piedra y Antonio Quintero por expandir mis conocimientos matemáticos.

Cristóbal Luque
Julio de 2009

Resumen

Desde siempre, un problema clásico en el campo de la Inteligencia Artificial ha sido la búsqueda de la forma de predecir comportamientos a partir de una base de datos que modeliza tales comportamientos. Un claro ejemplo son las Series Temporales, que buscan representar mediante medidas el comportamiento de un fenómeno a lo largo de un periodo de tiempo. Otro ejemplo clásico ha sido la predicción de la cotización de las acciones en bolsa, bién tratada como serie temporal, o bien en función de ciertos medidores.

El problema de la mayoría de los algoritmos de aprendizaje automático radica en su búsqueda de una aproximación global a estos problemas de predicción, es decir, buscan un modelo construido sobre todo el conjunto de patrones para predecir cualquier patrón. Esta tesis parte del planteamiento de que esta aproximación no es la más acertada, dado que no todos los patrones presentan las mismas características. Como ejemplo, un sistema de predicción de mareas que trate todos los patrones que representan las medidas del nivel del agua a lo largo de un año por igual nunca podrá ser tan acertado como uno que separe los patrones en conjuntos según la época del año, realice un aprendizaje para cada grupo, y posteriormente, según el grupo al que pertenezca, realice su labor de predicción correspondiente.

Así pues, nuestro objetivo ha sido la búsqueda de un algoritmo inteligente que no sólo sea capaz de aprender a predecir, sino también a buscar y clasificar las peculiaridades de cada subconjunto de datos, descargando esta tarea del investigador que quiera usar el algoritmo. La potencia de los algoritmos desarrollados en esta tesis se basan en la búsqueda y aprendizaje de y sobre estos subjuntos especiales. Esto nos permite incluso buscar comportamientos anómalos de los datos, y realizar reglas de predicción para ellos, lo cual es de vital importancia a la hora de predecir catástrofes.

En esta tesis se han desarrollado 2 algoritmos distintos, pero basados en la misma idea para el objetivo presentado. El primero está basado en las premisas de Packard sobre la predicción de sistemas dinámicos. La segunda es una idea completamente nueva, buscando una forma distinta de mejorar la primera aproximación. Ambos algoritmos se han usado sobre los mismos conjuntos de datos, obtenidos de campos completamente diferentes (series temporales artificiales, series reales, datos de bolsa, etc.), con unos resultados que mejoran, en la mayoría de los casos, los resultados de otros algoritmos clásicos avanzados de aprendizaje automático.

Abstract

A classic problem in the Artificial Intelligence area has been the prediction of behaviors using a data set modelling those behaviors. As examples, we can consider Time Series, which represent with measures the behavior of a certain phenomenon along a time period. Another classic example is the price of the stocks in the stock market, considered as time series or as a function for certain parameters too.

The main problem of most machine learning algorithms focuses on their search of a global approach to those prediction problems. That means, they try for create a model over the pattern set to predict another pattern. This is based on the idea that this approach is not the most indicated for all the problems, because not all the patterns present the same characteristics. For example, a system for tide forecasting that considers in the same category all the patterns, will not be as good as another system that separates the patterns into different sets representing different seasons, and then use those sets separately to extract the information and make predictions.

Thus, our objective has been the search of intelligent algorithms that are not only able to predict, but also to find and classify the characteristics of each data subset, thus avoiding that task to the researcher. The advantage of the algorithms developed in this thesis are the abilities to search and learn on special subsets of the data set. This also allows the algorithms to find abnormal behaviors in the data, making different predictions for them. That is a matter of utter importance to predict catastrophes.

In order to achieve our objective, two different algorithms have been developed and both are based on the same idea. The first one is based on the ideas of Packard about prediction on dynamical systems. Improving the first approach, we have developed the second one. Both algorithms have been tested on the same data sets, obtained from different domains (artificial time series, real time series, stock market, etc.) producing results that, in most cases, improve the results of other classic and advanced automatic learning algorithms.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El campo del aprendizaje automático, incluido dentro del área de la Inteligencia Artificial, tiene que ver con la construcción de programas de ordenador que mejoren con la experiencia de forma automática. En los últimos años se han desarrollado aplicaciones de aprendizaje automático que han tenido gran importancia práctica, pero también ha habido avances en la teoría y en los algoritmos que constituyen los fundamentos de este campo.

Para diseñar una aplicación de aprendizaje automático es necesario elegir bien el tipo de 'experiencia', es decir el tipo de ejemplos de entrenamiento, la función objetivo que debe ser aprendida, la representación para esta función, y un algoritmo para aprender la función objetivo a partir de los ejemplos de entrenamiento. El aprendizaje implica una búsqueda en un espacio de hipótesis para encontrar la hipótesis que mejor se adapte a los ejemplos de entrenamiento disponibles y a otras restricciones incluidas a priori. Existen muchos métodos de aprendizaje que buscan en los espacios de hipótesis; se pueden utilizar reglas simbólicas, árboles de decisión, redes de neuronas, etc.

Desde siempre, un problema clásico en el campo de la Inteligencia Artificial ha sido la búsqueda de la forma de predecir comportamientos a partir de una base de datos que modeliza tales comportamientos. Estos tipos de problemas son conocidos como dominios de regresión. La forma de modelizar estos datos es en forma de puntos en un espacio $n + 1$ dimensional, donde n es el número de variables del problema. Así pues, si las variables del problema son llamadas $x_1, ..., x_n$, y el valor que queremos predecir es y, el objetivo del problema es obtener una función f de forma que

$$f(x_1, ..., x_n) = y$$

y así poder obtener valores de y para cualquier posible valor de las variables.

Un campo dentro de los dominios de regresión son las Series Temporales, que buscan representar mediante medidas el comportamiento de un fenómeno a lo largo de un periodo de tiempo. Así pues, las variables de entrada para dicho problema son los instantes de tiempo anteriores. Como ejemplos de fenómenos que pueden modelizarse mediante series temporales tenemos, el nivel de la marea, la cotización de las acciones en bolsa, el consumo eléctrico, la velocidad de la sangre en el torrente sanguineo, etc.

Como métodos para afrontar la predicción de series temporales se han usado principalmente regresiones, sistemas dinámicos no lineales, redes neuronales, y en menor medida, algoritmos evolutivos.

Los **Algoritmos Evolutivos** constituyen un método práctico de optimización de una función objetivo. La tarea de optimización consiste en, dados los parámetros $(x_1, ..., x_n)$ de la función objetivo f, encontrar los valores $(\bar{x}_1, ..., \bar{x}_n)$ tales que hacen mínimo (o máximo, según sea el objetivo) el valor de la función en dicho punto, $f(\bar{x}_1, ..., \bar{x}_n)$.

Los Algoritmos Evolutivos tratan de imitar los procesos relacionados con la evolución natural, y principalmente la selección y la generación de nuevos genotipos para los descencientes de los individuos de la población.

1.1. Objetivos de la tesis

El problema de la mayoría de los algoritmos de aprendizaje automático radica en su búsqueda de una aproximación global a estos problemas de predicción, es decir, buscan un modelo construido sobre todo el conjunto de patrones para predecir cualquier patrón. Esta tesis parte del planteamiento de que esta aproximación no es la más acertada, dado que no todos los patrones presentan las mismas características. Como ejemplo, en un sistema de predicción de mareas que trate todos los patrones que representan las medidas del nivel del agua a lo largo de un año por igual nunca podrá ser tan acertado como uno que separe los patrones en conjuntos según la época del año, realice un aprendizaje para cada grupo, y posteriormente, según el grupo al que pertenezca, realice su labor de predicción correspondiente. La figura 1.1 muestra gráficamente un ejemplo en el que varias funciones simples (regresiones, o en este caso rectas) producen una aproximación más acertada que una única función de un orden de complejidad mucho mayor que una simple regresión.

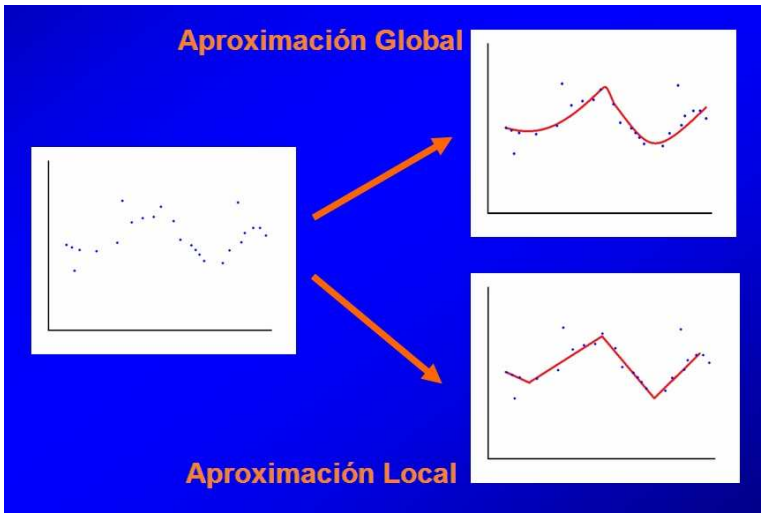

Figura 1.1:
Aprendizaje Global vs aprendizaje Local

Así pues, nuestro objetivo es la búsqueda de algoritmos inteligentes que no sólo sean capaces de aprender a predecir, sino también de buscar y clasificar las peculiaridades de cada subconjunto de datos, descargando esta tarea del investigador que quiera usar dichos algoritmos. La potencia de los algoritmos desarrollados en esta tesis se basan en la búsqueda y aprendizaje sobre estos subjuntos especiales. Esto nos permite incluso buscar comportamientos anómalos de los datos, y realizar reglas de predicción para ellos, lo cual es de vital importancia a la hora de predecir catástrofes que cualquier sistema de predicción estándar podría clasificar como un comportamiento normal.

También se requiere que nuestros algoritmos tengan la característica de detectar cuándo se va a producir un comportamiento tan extraño que no pudiese ser predicho a partir de los datos basados en sucesos anteriores. En estos casos, lo mejor es producir un mensaje de error y no devolver una predicción que estuviese muy alejada de la real, y produjese un desequilibrio en el sistema.

En esta tesis se busca desarrollar algoritmos basados en la misma idea para el objetivo presentado. El primero está basado en las premisas de Packard sobre la predicción de sistemas dinámicos. La segunda buscará una forma distinta de mejorar la primera aproximación. Ambos algoritmos se han usado sobre los mismos conjuntos de datos, obtenidos de campos completamente diferentes (series temporales artificiales, series reales, datos de

bolsa, etc.), con unos resultados que mejoran en la mayoría los resultados de otros algoritmos clásicos avanzados de aprendizaje automático. En los resultados obtenidos, puede observarse que cuando los algoritmos presentados no superan a los algoritmos con los que se comparan, suelen obtener resultados equivalentes.

1.2. Organización de la tesis

El resto de los capítulos de esta memoria están organizados del siguiente modo: En el capítulo 2 se revisan los dos temas que están directamente relacionados con la presente tesis, las series temporales y la aplicación de los algoritmos genéticos en el campo de la predicción. Se describen los principales trabajos relacionados con ellos y se finaliza con una discusión sobre los principales inconvenientes de estas técnicas.

En el capítulo 3 se exponen los objetivos de la tesis, que pretenden solucionar los inconvenientes detectados en las técnicas antes mencionadas.

En el capítulo 4, se describe el primer algoritmo propuesto.

En el capítulo 5, se analiza el segundo algoritmo.

En el capítulo 6, se explicarán qué dominios han sido seleccionados para probar ambos algoritmos. Así mismo, se contrastarán los resultados de ambos con los de otros algoritmos de aprendizaje automático.

Por último, en el capítulo 7 se muestran las principales conclusiones derivadas de este trabajo, y se plantean líneas futuras de investigación.

Capítulo 2

Estado del arte

2.1. Series Temporales

Una serie temporal o cronológica es una secuencia de datos, observaciones o valores, medidos en determinados momentos del tiempo, espaciados entre sí de manera uniforme y por tanto ordenados cronológicamente. El análisis de series temporales comprende métodos que ayudan a interpretar este tipo de datos, extrayendo información representativa, tanto referente a los orígenes o relaciones subyacentes como a la posibilidad de extrapolar y predecir su comportamiento futuro.

De hecho uno de los usos más habituales de las series de datos temporales es su análisis para predicción y pronóstico. Por ejemplo de los datos climáticos, de las acciones de bolsa, o las series pluviométricas. Resulta difícil imaginar una rama de las ciencias en la que no aparezcan datos que puedan ser considerados como series temporales.

2.1.1. Componentes

El análisis más clásico de las series temporales se basa en la suposición de que los valores que toma la variable de observación es la consecuencia de cuatro componentes, cuya actuación conjunta da como resultado los valores medidos; estos componentes son:

- La tendencia secular o regular, indica la marcha general y persistente del fenómeno observado, es una componente de la serie que refleja la evolución a largo plazo. Por ejemplo, la tendencia creciente del índice de reciclado de basuras en los países desarrollados, o el uso creciente de internet en la sociedad, independientemente de que en un mes concreto

en un país, por determinadas causas, haya una bajada de la utilización de internet.

- Variación estacional. Es el movimiento periódico de corto periodo, se trata de una componente causal debida a la influencia de ciertos fenómenos que se repiten de manera periodica en una año (las estaciones), una semana (los fines de semana) o un día (las horas puntas) o cualquier otro periodo, recoge las oscilaciones que se producen en esos períodos de repetición.

- Variación cíclica. Es el componente de la serie que recoge las oscilaciones periódicas de amplitud superior a un año. movimientos normalmente irregulares alrededor de la tendencia, en las que a diferencia de las variaciones estacionales, tiene un período y amplitud variables, pudiendo clasificarse como cíclicos, cuasicíclicos o recurrentes.

- Variación aleatoria, accidental, de carácter errático, también denominada residuo, no muestran ninguna regularidad, debidos a fenómenos de carácter ocasional como pueden ser tormentas, terremotos, inundaciones, huelgas, guerras, avances tecnológicos etc.

2.2. Predicción de Series Temporales

La predicción de series temporales es un problema complejo, consistente en predecir el comportamiento de una serie de datos a partir de la información extraída de la secuencia. Asi pues, definimos una serie temporal con un conjunto de n datos $\{S_i\}_{1 \leq i \leq n}$. El objetivo del problema será predecir S_j para $j > n$.

Muchos fenómenos físicos y artificiales (como por ejemplo, los niveles de marea [Zaldivar et al., 2000], consumo de agua [Luque et al., 2007], índices de valores en bolsa, velocidad de la sangre en el torrente sanguíneo [Mackey and Glass, 1977], mortalidad en terremotos[Gutiérrez et al., 2005], etc.) pueden modelarse con series temporales, lo cual hace el problema de la predicción tan complejo como interesante. El proceso de creación de una serie temporal es tan simple como archivar las medidas del fenómeno a predecir, para períodos de tiempo iguales: por ejemplo la cantidad de litros por metro cuadrado caídos a lo largo de una semana en una zona.

Algunas aproximaciones al problema prefieren usar datos adicionales a la propia serie, a la hora de predecir: por ejemplo, a la hora de predecir el nivel de marea, si además de la propia serie temporal contamos con la medida de

la presión barométrica en ese instante de tiempo, la predicción tenderá a ser más acertada. Para este trabajo, nos hemos centrado en el estudio de las series temporales en sí mismas, aunque los sistemas investigados admiten la entrada de datos adicionales a la serie.

Hay muchos métodos para obtener buenas predicciones, pero en la mayoría de los casos, estos métodos sólo buscan una aproximación general al comportamiento de la serie. El problema radica en que las series temporales tienen generalmente comportamientos locales que no permiten un buen nivel de predicción al usar una aproximación global. Este trabajo presenta métodos para encontrar comportamientos locales capaces de hacer predicciones a estos niveles, y en conjunto alcanzar una mejor predicción total.

Definimos horizonte de predicción como la cantidad de instantes de tiempo entre el último instante usado para predecir y el instante a predecir. En el caso de que usasemos $\{S_i\}_{1\leq i\leq n}$ para predecir S_{n+T}, el horizonte de predicción sería T. A la hora de realizar la predicción, hay tres técnicas básicas.

- La primera consiste en entrenar un modelo que prediga la serie para cualquier instante de tiempo posterior. Es decir, a partir de $\{S_i\}_{1\leq i\leq n}$ creamos un sistema F_1 de forma que

$$\forall j > n,\ \hat{S}_j := F_1(S_1, S_2, ..., S_{n-1}, S_n)$$

  .

- El segundo método consiste en entrenar un sistema F_2 para que prediga para un horizonte de predicción 1. En este caso:

$$\hat{S}_{n+1} := F_2(S_1, S_2, ..., S_{n-1}, S_n)$$

$$\hat{S}_{n+2} := F_2(S_2, S_3, ..., S_{n-1}, S_n, \hat{S}_{n+1})$$

  y en general

$$\hat{S}_{n+k} := F_2(\hat{S}_k, ..., \hat{S}_{n+k-2}, \hat{S}_{n+k-1})$$

  Habitualmente, este método es el menos recomendado, ya que para cada incremento en el instante a predecir se va acumulando un error cada vez mayor.

- El tercer método consiste en realizar varios entrenamientos por separado para un mismo sistema, según el horizonte de predicción. En este caso, para el sistema F_3, usamos la serie $\{S_i\}_{1\leq i\leq n}$ para entrenar m sistemas F_3^1,... ,F_3^m, de forma que

$$\hat{S}_{n+1} := F_3^1(S_1, S_2, ..., S_{n-1}, S_n)$$

$$\hat{S}_{n+2} := F_3^2(S_1, S_2, ..., S_{n-1}, S_n)$$

y en general

$$\hat{S}_{n+k} := F_3^k(S_1, S_2, ..., S_{n-1}, S_n)$$

para $k \leq m$. De los tres métodos, este es el más costoso computacionalmente.

Para la mayoría de los casos, el tercer método suele obtener mejores resultados, y por tanto este será el elegido para los métodos investigados.

2.2.1. Transformación de una Serie Temporal en Patrones

La mayoría de los sistemas de aprendizaje automático trabajan a partir de patrones. Los patrónes representan ejemplos, a partir de cuales se realiza el proceso de aprendizaje, o bien se comprueba que el aprendizaje ha sido correcto, si el patrón es de validación. Si queremos realizar un aprendizaje de una función de s parámetros de entrada y t parámetros de salida, un patrón será un vector de $s + t$ coordenadas de la forma:

$$(x_1, x_2, ..., x_s, y_1, y_2, ..., y_t)$$

donde los x_i serán los valores de entrada, y los y_j los valores de salida, para $1 \leq i \leq s$, y $1 \leq j \leq t$.

Para la construcción de patrones, debemos definir el concepto de ventana temporal. La ventana temporal se define como la cantidad de instantes de tiempo consecutivos que se usan para construir un patrón. Así pues, con una ventana temporal de 5 instantes y un horizonte 3, se podrán construir los siguientes patrones a partir de la serie temporal $\{S_i\}_{1 \leq i \leq n}$ (los datos de salida del patrón están en negrita):

$$(S_1, S_2, S_3, S_4, S_5, \mathbf{S_8})$$

$$(S_2, S_3, S_4, S_5, S_6, \mathbf{S_9})$$

$$...$$

$$(S_{n-7}, S_{n-6}, S_{n-5}, S_{n-4}, S_{n-3}, \mathbf{S_n})$$

En general, para una ventana de $h > 0$ instantes y un horizonte de $k > 0$, se puede construir el patrón:

$$(S_i, S_{i+1}, ..., S_{i+h-1}, \mathbf{S_{i+h-1+k}})$$

siempre y cuando $i > 0$, y $(i + h - 1 + k) \leq n$.

En general no es necesario usar todos los datos de la ventana temporal. Por ejemplo, para la serie temporal de Mackey-Glass, en [Yingwei et al., 1997] se usa una ventana temporal de 18 instantes, pero de estos 18 sólo se usan 4 instantes, en concreto los 0, 6, 12 y 18 ultimos instantes. Con esto se reduce la complejidad del sistema a entrenar.

De los sistemas con los cuales contrastamos resultados, hay algunos que sólo pueden producir salidas en rangos limitados, como es el caso del perceptrón multicapa, que sólo pueden producir salidas con valores entre -1 y 1. Así pues, la serie temporal debe ser normalizada. Para ello debemos encontrar el máximo y el mínimo de la serie. Así pues, definimos:

$$S^M := max_{1 \leq i \leq n}\{S_i\}$$

$$S^m := min_{1 \leq i \leq n}\{S_i\}$$

La normalización se puede hacer en cualquier intervalo que esté dentro del rango de salidas. Para este trabajo se eligió la normalización en el rango $[0, 1]$, que responde a la siguiente función:

$$norm(x) := \frac{x - S^m}{S^M - S^m}$$

A partir de esto, definimos la normalización de un patrón $P = (x_1, x_2..., x_p)$ como:

$$norm(P) := (norm(x_1), norm(x_2), ..., norm(x_p))$$

Para los experimentos realizados en esta tesis, todos los patrones serán normalizados[1].

En algunas ocasiones, también se puede requerir la reordenación de los patrones. Como ejemplo, supongamos que tenemos los datos de un fenómeno físico medido a lo largo de un año, y queremos entrenar un sistema de apredizaje con los patrones construídos a partir de estos datos. Si a continuación dividimos el conjunto de datos de forma secuencial, es decir, el primer 80 % de los patrones los destinamos a entrenamiento, y el 20 % restante a validación, estaríamos cometiendo un error, ya que la estructura de los datos durante una serie de meses consecutivos del año puede ser muy distinta de la misma en el resto de los meses, imposibilitando así la capacidad de generalización de los sistemas de aprendizaje automático. Así pues, en estos casos se suele requerir una reordenación de los patrones antes de realizar

[1] Los algoritmos desarrollados en esta tesis no requieren que los datos sean normalizados, pero dado que algunos de los implementados en WEKA sí los requerían, se optó por la normalización

su división en conjuntos de entrenamiento o validación. Normalmente, la reodenación a la que más se suele recurrir, es la reordenación aleatoria.

Para el ejemplo dado, si se tuviesen más datos referentes a otros años, realmente no hubiese sido necesario reordenar los patrones, ya que habría información suficiente para el aprendizaje.

2.2.2. Formas de afrontar la predicción de Series Temporales

En la predicción de Series Temporales, las técnicas más utilizadas son ARMA (AutoRegressive Moving Average), ARIMA [Box et al., 1976] y regresiones [Papalexopoulos and Hesterberg, 1990]. En [Pulido et al., 2005] se aplicaron técnicas de Sistemas de Identificación. Para estos sistemas, la serie se considera una señal con un periodo T, que se modela con un modelo polinomial paramétrico ARMAX. La debilidad de estos sistemas con respecto a las Series Temporales en general es su reducida capacidad de adaptación, sobre todo a problemas no lineales. Evitar este escollo ha motivado la búsqueda en este campo de investigación de otras herramientas de exigencias computacionales más costosas [Cleveland and Devlin, 1988] y del área de la Inteligencia Artificial, como son las Redes Neuronales [Park et al., 1991] y los Sistemas Expertos [Rahman and Hazim, 1993].

La volatilidad estocástica (definida como la varianza condicional de la serie) es una característica inherente a las series temporales financieras. En general, no suele ser constante y en consecuencia los modelos tradicionales para series temporales no suelen ser adecuados para modelar series temporales financieras. Engle [Engle, 1982] introdujo una nueva clase de procesos estocásticos llamados modelos ARCH, en los cuales la varianza condicionada a la información pasada no es constante, y depende de los datos anteriores. Posteriormente, Bollerslev [Bollerslev, 1986] generalizó los modelos ARCH al proponer los modelos GARCH en los cuales la varianza condicional depende no sólo de los cuadrados de las perturbaciones, sino además, de las varianzas condicionales de períodos anteriores. A continuación se desarrollaron los modelos EGARCH[Nelson, 1991], en los cuales se crea un modelo que no se comporta de manera simétrica para perturbaciones positivas y negativas de la varianza condicional, como sucede en los modelos GARCH; expresando otro rasgo de la volatilidad: su comportamiento asimétrico frente a las alzas y bajas de los precios de un activo financiero. En la última década se han seguido publicando trabajos sobre modelos de volatilidad y sus aplicaciones [Poon and Granger, 2003, Hansen and Lunde, 2006, Casas and Cepeda, 2008].

La regresión local[Cleveland, 1979, Cleveland and Devlin, 1988] (LOESS

o LOWESS - LOcally WEighted Scatterplot Smoother), es uno de muchos métodos modernos de construcción de modelos basados en los clásicos, como la regresión lineal y no lineal. Este método se basa en el cálculo de aproximaciones locales para cada uno de los datos de entrada $\{\vec{x_1}, ..., \vec{x_n}\}$. Dichas aproximaciones son polinomios de grado muy bajo (un grado 2 suele ser más que suficiente) calculados mediante mínimos cuadrados, y que sólo tienen en cuenta de forma ponderada los k datos más cercanos al dato $\vec{x_i}$ en que se calculan. La influencia de los datos cercanos es ponderada mediante una función de peso $W(u)$. Dicha función de peso sólo debe cumplir una serie de requisitos (especificados en [Cleveland, 1979]), aunque la más comunmente usada es $W(u) = (1 - u^3)^3$ para $0 \leq u < 1$, y 0 para el resto de valores de u. Así pues, si estamos calculando la aproximación local en el punto $\vec{x_i}$, y siendo D la distancia de $\vec{x_i}$ al $k-$ésimo punto más cercano (donde la distancia usada, d, puede ser la euclídea o la que se prefiera), el peso asignado al punto $\vec{x_j}$ será:

$$w_i(\vec{x_j}) = W(d(\vec{x_i}, \vec{x_j})/D)$$

la cual devuelve un valor cercano a 1 para los datos más próximos a $\vec{x_i}$, un valor casi nulo para los que tengan un orden de cercanía de $k-1, k-2$, y de 0 para los datos que se encuentren a una distancia mayor o igual a D. Las aplicaciones y propiedades del sistema LOESS han sidos discutidas por diferentes autores, y entre otros [Gray and Thomson, 1990, Fan and Gijbels, 1996, Dagum and Luati, 2002].

Las redes neuronales artificiales destacan como métodos eficaces para la predicción de series temporales [Platt, 1991, Valls et al., 2007]. Otros trabajos usan métodos de aprendizaje para seleccionar automáticamente los patrones más apropiados para el entrenamiento dependiendo del ejemplo que se va a predecir. Este método de entrenamiento utiliza una estrategia de aprendizaje retardado, en el sentido de que construye aproximaciones locales centradas alrededor de la nueva muestra. En su artículo, Galván et al. [Galván et al., 2001] aplican su método tanto a la serie de Mackey-Glass, como a la serie temporal del nivel de agua de la laguna de Venecia.

Otras aproximaciones clásicas pueden encontrarse en [Zaldivar et al., 2000], donde se hace un análisis de series temporales usando teoría de sistemas dinámicos no lineales así como modelos basados en redes de neuronas multicapa. Estos modelos son aplicados a unas medidas tomadas del nivel de la marea en la laguna de Venecia a lo largo de los años 1980-1994.

Packard sugiere otra forma de afrontar el problema de la predicción de sistemas dinámicos [Meyer and Packard, 1992, Mitchell, 1996, Packard, 1990], usando algoritmos genéticos [Holland, 1975] para generar reglas de predic-

ción sobre una serie temporal, tal y como se verá en la sección siguiente.

Trabajos más recientes dentro de los Algoritmos Evolutivos usan GEP (Gene Expression Programming) [Zuo et al., 2004] y ya fuera del área de la Inteligencia Artificial, funciones matemáticas de base cardinal B-Spline [Wei and Billings, 2006].

Una función de base cardinal de orden m se calcula de forma recursiva mediante la ecuaciones 2.1 y 2.2, que fueron definidas en [Chui, 1992].

$$N_m(x) = \frac{x}{m-1} N_{m-1}(x) + \frac{m-x}{m-1} N_{m-1}(x-1), m \geq 2 \tag{2.1}$$

$$N_1(x) = \begin{cases} 1 & \text{si } x \in [0,1) \\ 0 & \text{eoc} \end{cases} \tag{2.2}$$

El método para predecir series temporales consiste en aproximar una función $f(x)$ tal que $f_i(x) = x_i$ encontrando los coeficientes $\{c_k^m\}_{k \in C} \in \ell^2(\mathbb{Z})$ tales que

$$f(x) = \sum_{k \in C} c_k^m s^{j/2} N_m(s^j x - k)$$

En teoría, C podría ser igual a $\mathbb{Z}$, pero ciertos teoremas matemáticos nos garantizan que siendo C un subconjunto finito de $\mathbb{Z}$ obtenemos una aproximación bastante cercana a $f(x)$.

2.2.3. Clasificación contra Predicción

Evidentemente, para un sistema capaz de realizar aproximaciones de funciones, la tarea de clasificación es trivial. Si tenemos un conjunto de patrones P y un conjunto de prototipos $S = \{S_1, ..., S_n\}$, basta con aproximar la función vectorial $\bar{f}(p)$, definida en la ecuación 2.3.

$$\bar{f}(p) = \bar{e}_i \text{ si } p \in S_i \tag{2.3}$$

donde $\bar{e}_i$ es el i-ésimo vector canónico n-dimensional. Es decir:

$$\bar{e}_1 = (1, 0, 0, ..., 0)$$

$$\bar{e}_2 = (0, 1, 0, ..., 0)$$

$$\bar{e}_3 = (0, 0, 1, ..., 0)$$

$$...$$

$$\bar{e}_{n-1} = (0, 0, 0, ..., 1, 0)$$

$$\bar{e}_n = (0, 0, 0, ..., 1)$$

Así pues, si por ejemplo el patrón q pertenece a la clase S_2, entonces $\bar{f}(q) = (0, 1, 0, ..., 0)$. Otra posible opción para el sistema aproximador de funciones sería aproximar la función $g(p)$ definida como

$$g(p) = \bar{f}(p) \cdot (1, 2, 3, ..., n)$$

donde $\cdot$ representa el producto escalar de dos vectores. O lo que es lo mismo

$$g(p) = i \text{ si } p \in S_i$$

En cualquier caso, esta segunda aproximación no tiene por qué dar tan buen resultado como la primera, ya que la reducción de la dimensionalidad del problema puede eliminar información. Por ejemplo, con la segunda aproximación, si un patrón perteneciente a la clase i se clasificase en la clase $i-1$ produciría un error más bajo que si se clasificase en la clase $i-2$, con lo cual hay dos problemas:

- El error de clasificación depende del orden en que suministremos la lista de clases $S = \{S_1, ..., S_n\}$, lo cual puede influir en los sistemas basados en descenso de gradiente.
- En teoría, un patrón mal clasificado debería producir una medida de error independiente de la clase a la que erróneamente se hubiese mandado. Si está mal clasificado, lo está independientemente de a donde se mande, y no más o menos mal clasificado[2]

2.3. Algoritmos Evolutivos

Los algoritmos evolutivos son técnicas englobadas dentro del campo de la inteligencia artificial. Dichas técnicas se aplican principalmente a problemas de optimización, y su principal característica consiste en tratar las posibles soluciones al problema como individuos que evolucionan a lo largo de iteraciones (también llamadas "generaciones"). Este conjunto de algoritmos destacan especialmente cuando el problema de optimización a tratar posee un gran espacio de búsqueda, no son lineales, poseen gran cantidad de máximos o mínimos locales, o en general no son resolubles por las técnicas estándar de búsqueda (como descenso del gradiente).

[2]No siempre es así, y es posible que existan algunos dominios en los cuales puede interesarnos tener una gerarquía de clases. Es posible que mandar un patrón a una clase sea erróneo, pero menos que mandarlo a una tercera clase.

Siguiendo la terminología de la teoría de la evolución, los objetos que representan las soluciones al problema se denominan individuos (y su representación cromosomas), y el conjunto de éstos, población. Nuevos individuos son producidos a partir de los ya existentes en la población mediante los operadores genéticos, principalmente el sobrecruzamiento, que consiste en la creación de un nuevo individuo a partir de la mezcla de la información de dos o más individuos; otro operador es el de mutación, que es un cambio aleatorio en la información de los individuos. La selección, consistente en la elección de los individuos que sobrevivirán y formarán la siguiente generación. Dado que los individuos que representan las soluciones más adecuadas al problema tienen más posibilidades de sobrevivir, la población va mejorando gradualmente. Esta selección se produce a partir de una función de aptitud, o *fitness*. La funciónde fitness tiene como objetivo guiar la evolución de forma que el más apto tendrá mejor valoración de fitness, o lo que es lo mismo, un invididuo será mejor solución cuanto más alto sea su valor de fitness.

Los tres principales paradigmas dentro de los algoritmos evolutivos son:

- Algoritmos Genéticos
- Estrategias Evolutivas
- Programación Genética

Para este trabajo de investigación se han recurrido las técnicas basadas en **Algoritmos Genéticos** y **Estrategias Evolutivas**, que se detallarán en las secciones siguientes. Se puede encontrar más información sobre los algoritmos evolutivos en [Fogel, 1994].

2.3.1. Algoritmos Genéticos

El precursor de este campo de investigación fue **John Henry Holland**, que durante los años 70 estuvo investigando la forma de aplicar las teorías de la evolución al campo de la ciencia computacional [Holland, 1975]. Su idea se basaba en usar poblaciones de individuos que, mediante cadenas de ceros y unos, representaban soluciones a cierto problema. A partir de estas poblaciones, y mediante técnicas que simulaban el sobrecruzamiento, mutación y la selección natural, se buscaba obtener mejores soluciones.

Los algortimos genéticos, siguen el esquema detallado en el Algoritmo 1.

Dentro de los **Algoritmos Genéticos**, a su vez hay dos interesantes perspectivas: la de Pittsburgh [Smith, 1980, Smith, 1983] y la de Michigan [Holland and Reitman, 1977, Booker, 1982]. En la primera, que es la

Algorithm 1 Esquema de los Algoritmos Genéticos

```
Inicializar población
mientras no condicion-de-parada
        engendrar individuos
        mutar
        seleccionar
```

más usada, cada individuo representa una posible solución al problema. La población por tanto, representa un conjunto de soluciones. En la segunda, cada individuo representa sólo una parte de la solución, que vendrá definida por toda la población (que en este caso representará una única solución). Cada una de estas perspectivas tiene su ventaja, como el hecho de que las evaluaciones son más rápidas y los individuos ocupan menos espacio en memoria, para la perspectiva Michigan. En el otro lado está el hecho de que la perspectiva de Pittsburgh nos da todo un conjunto de soluciones, lo cual siempre es más ventajoso que obtener sólo una. Esta perspectiva es imprescindible para los algoritmos genéticos multiobjetivo. En cualquier caso, y dada la localidad que buscamos para esta tesis, no hemos decantado por Michigan.

Hoy en día, los algoritmos genéticos se aplican no sólo a la optimización de funciones, sino a campos tan diversos como por ejemplo, la criptografía [Isasi and Hernández, 2004, Hernández and Isasi, 2004, Nedjah et al., 2007], teoría de juegos [Axelrod, 1997, Mochón et al., 2005], economía y finanzas [Anthony and Jennings, 2002, Mochón et al., 2008, Quintana et al., 2005], arte y creatividad musical [Romero and Machado, 2007, Miranda and Biles, 2007, Sáez et al., 2004], diseño [Sáez et al., 2005], biología [Kuzmanovski et al., 2005], ... etc.

2.3.2. Estrategias Evolutivas

Las Estrategias Evolutivas fueron creadas por Rechenberg y Schwefel en la Universidad Técnica de Berlin [Schwefel, 1965], y su primer campo de investigación fue el diseño de túneles de viento.

Las dos principales diferencias de las estrategias evolutivas con respecto a los otros algoritmos evolutivos son que su codificación se hace sobre vectores de números reales y que en las estrategias evolutivas únicamente se usa el operador de mutación [Fogel, 1994, Bäck et al., 1991], es decir, no hay intercambio de información entre padres para tener un descendiente, lo cual efectivamente ocurre en otros paradigmas como los algoritmos genéticos o la

programación genética [Fogel, 1994, Holland, 1975, Mitchell, 1996]. En las estrategias evolutivas el proceso de engendrar produce los λ hijos a partir de los μ padres mediante mutación. Hay varias formas de producir estos descendientes por mutación de los datos del padre, de las cuales detallamos las principales, pero todas ellas exigen que el individuo mantenga una información adicional referente a la amplitud de la mutación, al contrario que en los demás **Algoritmos Evolutivos**. La primera forma de mutar consistente en alterar aleatoriamente la información del progenitor según una función aleatoria gausiana. Matemáticamente escribimos este operador de mutación del elemento x como $N(x, v)$, es decir, la mutación del valor de x es una variable aleatoria normal de media x y varianza v. A su vez esta varianza también suele ser un parámetro que el descendiente hereda y muta según la fórmula $v' = ve^{N(0,A)}$, donde A es una constante que se suele ajustar para cada problema. Esta varianza viene a representar cuán cerca está ese elemento en concreto de la solución: una varianza pequeña producirá, la mayoría de las veces, alteraciones pequeñas, y una varianza grande, alteraciones grandes, con lo cual es lógico que este parámetro se herede.

La segunda forma es la llamada regla 1/5, basada en los postulados de Rechenberg [Bäck et al., 1991], que asegura que en una estrategia (1+1) la proporción de descendientes que sustituyen al padre debe ser 1/5. Si es mayor que 1/5 debemos incrementar la varianza, y si es menor, decrementarla. En este método la varianza se mantiene fija, y si la función a optimizar tiene n variables, cada n generaciones observamos la proporción de sustituciones del padre que se han producido en las últimas $10n$ generaciones. Si esta proporción es mayor que 1/5, la varianza se multiplica por una constante de incremento c_i; si es menor que 1/5, la varianza se multiplica por una constante de decremento c_d. Si la proporción es exactamente 1/5, entonces se mantiene la varianza. Como constantes se suelen usar $c_d = 0{,}82$ y $c_i = 1/0{,}82$

A la hora de seleccionar la población hay dos alternativas: $(\mu + \lambda)$ y (μ, λ). En la $(\mu + \lambda)$ μ padres producen $\lambda \geq \mu$ hijos, y entre la población total de $\mu + \lambda$ individuos se seleccionan los μ mejores, que pasan a ser los padres de la siguiente generación. En la (μ, λ) μ padres producen $\lambda \geq \mu$ hijos, y de entre estos λ hijos se seleccionan los μ mejores, que pasan a ser los padres de la siguiente generación.

Se puede encontrar más información sobre estrategias evolutivas, en [Bäck et al., 1991, Bäck and Schwefel, 1992], así como algunas de sus aplicaciónes [Luque et al., 2004a].

2.4. Algoritmos para la predicción de Series Temporales

En la literatura se han desarrollado distintas aplicaciones del área de la inteligencia artificial al campo de las Series Temporales. Dentro de éstas, existen diferentes técnicas para dividir el espacio de entrada en regiones. Estas son, pues, las que más nos interesan y las que desarrollan una idea más cercana a la aproximación presentada en esta Tesis. El sistema **LVQ** [Somervuo and Kohonen, 1999] usa regiones de Voronoi para tareas de clasificación. Las **Redes de Base Radial** (Radial Basis Neural Networks - RBNN)[Moody and Darken, 1989] , en uno de sus métodos no supervisados clásicos, divide el espacio de entrada en regiones de Voronoi usando el algoritmo de K-medias [Lloyd, 1982, Macqueen, 1967] u otros algoritmos similares para determinar los centros de las funciones de base radial.

Las RBNN también poseen otras formas alternativas de determinar los centros de las funciones de base radial, modificando la posición de los centroides de forma supervisada para minimizar el error cuadrático medio de la salida. Esta idea es más cercana a la presentada en este trabajo: ajustar los centroides durante el proceso de minimización del error.

También se han utilizado Algoritmos Genéticos para evolucionar regiones de Voronoi en problemas de clasificación, tal como se expone en [Fernández and Isasi, 2008]. Packard [Packard, 1990] también usó algoritmos genéticos para afrontar el problema de predicción en sistemas dinámicos: en [Meyer and Packard, 1992, Mitchell, 1996], dicho autor presenta un nuevo enfoque en el cual se divide el espacio de entrada mediante reglas condicionales. Para el primer modelo presentado en esta Tesis, se trató de mejorar la idea de Packard, tal como se muestra en nuestros trabajos previos [Luque et al., 2004b, Quintana et al., 2005]

Otros trabajos también han usado la idea de construir modelos locales [Pavlidis et al., 2004, Vrahatis et al., 2002], usando el sistema de clustering k-windows.

A continuación, pasamos a describir brevemente todos estos algoritmos de particionado del espacio.

2.4.1. K-Medias

El algoritmo de K-medias es un algoritmo de "clustering", o lo que es lo mismo busca situar un número m de prototipos de forma que cada uno esté centrado en una acumulación de datos de entrada o patrones, que llamaremos $T = \{t_1, ..., t_s\} \subset \mathbb{R}^n$. El conjunto finito de prototipos

será $\varphi = \{P_1, ..., P_m\}$. Cada uno de estos prototipos define una región de Voronoi. Llamaremos V_i a la región definida por P_i, con $1 \leq i \leq m$. Un punto p del espacio de entrada pertenecerá a la región V_i, y solo a ella si

$$d(p, P_i) < d(p, P_j), \forall j \neq i, 1 \leq j \leq m$$

siendo d la distancia euclieda en el espacio $\mathbb{R}^n$, es decir, d es la función definida en la ecuación 2.4 para los puntos $\bar{x} = (x_1, ..., x_n)$ e $\bar{y} = (y_1, ..., y_n)$.

$$d(\bar{x}, \bar{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.4}$$

El algoritmo de K-medias necesita como entradas el número de prototipos que queremos usar, m, el número de interaciones máximo, M, y el conjunto de patrones $T = \{t_1, ..., t_s\} \subset \mathbb{R}^n$. Una vez suministrados estos datos, el algoritmo de K-medias sigue el siguiente esquema:

- PASO 1: Se inicializan aleatoriamente los m prototipos $P_1, ..., P_m$, e inicializamos a 0 el contador de iteraciones z. También fijamos una distancia suficientemente pequeña ϵ.
- PASO 2: Para cada prototipo P_i, buscamos los patrones de $T = \{t_1, ..., t_s\}$ que pertenezcan a su región de Voronoi correspondiente. Llamaremos a este subconjunto $T_i \subset T$, y por tanto estará definido por la ecuación 2.5.

$$T_i = \{t \in T | t \in V_i\} \tag{2.5}$$

- PASO 3: Para cada índice i, calculamos $\bar{P}_i$ como el centro de masas discreto del conjunto T_i. Es decir, si $T_i = \{q_1, ..., q_k\} \subset T$, y a su vez $q_l = (q_1^l, ...q_n^l)$, entonces el protipo $\bar{P}_i$ será calculado como el vector definido en la ecuación 2.6.

$$\bar{P}_i = (\frac{\sum_{i=1}^{k} q_1^i}{k}, ..., \frac{\sum_{i=1}^{k} q_n^i}{k}) \tag{2.6}$$

- PASO 4: Incrementamos en uno el contador de iteraciones z. Si $z = M$, o si para todo i tenemos que $d(P_i, \bar{P}_i) < \epsilon$, entonces hemos terminado. En caso contrario, asignamos a P_i el valor de $\bar{P}_i$ para cada i, y volvemos al PASO 2.

De esta forma obtenemos una serie de prototipos que representan a los patrones que pertenecen a su región de Voronoi.

2.4.2. K-Windows

El algoritmo de clustering k-windows [Vrahatis et al., 2002] trata de situar una ventana d-dimensional de forma que incluya todos los patrones que pertenezcan a cada prototipo, para cada uno de los prototipos.

El funcionamiento de este algoritmo es parecido al de K-medias salvo por el hecho de que usa hipercubos (o ventanas) en vez de regiones de Voronoi, y que todos los hipercubos no llegan a recubrir todo el espacio; en realidad, puede haber patrones que no estén dentro de ningúna ventana.

En un principio se seleccionan k puntos aleatoriamente. Las d-ventana iniciales, de un tamaño a, se centrarán en dichos puntos iniciales. A continuación, se identifican los patrones que pertenezcan a cada d-ventana. Posteriormente se calcula la media de los patrones pertenecientes a cada d-ventana, y dicho punto se asigna al centro de la d-ventana. Con esto, ajustamos mejor la posición de la d-ventana a los puntos que contiene. Este proceso se va reiterando hasta que no hay alteraciones en las posiciones de las d-ventanas (o hasta un máximo de iteraciones, o hasta que la variación de la posición de la d-ventana está por debajo de cierto umbral θ_v).

Una vez que la fase de movimiento de las d-ventanas ha terminado, viene una segunda fase en la cual estas se expanden de forma que capturen tantos patrones como les sea posible. Este alargamiento se hace por separado para cada dimensión de la d-ventana. Cada d-ventana se alarga en un porcentaje de θ_e/l en cada dimensión, donde θ_e es un valor definido por el usuario, y l se mantiene del número de alargamientos favorables previos. Despues de que se produzca el alargamiento unidimensional, la ventana de mueve de la forma anteriormente descrita. Después de este nuevo proceso de movimiento, se calcula el incremento proporcional de patrones que ha ganado esta d-ventana. Si dicha proporción no supera el umbral de cobertura definido por el usuario, θ_c, se rechazan los cambios (movimientos y alargamientos) para esta d-ventana, y se devuelve a su estado anterior. En caso contrario, todos estos cambios se aceptan. Si este alargamiento se acepta para la dimensión $d' \geq 2$, entonces, para todas las dimensiones d'' tales que $d'' < d'$ se realiza de nuevo el proceso de alargamiento, asumiendo como posición inicial la actual de la ventana. El proceso termina cuando el alargamiento en alguna dimensión no produce un incremento proporcional del número de patrones contenidos en la ventana mayot que el umbral θ_c.

2.4.3. Redes de Cuantización Vectorial (LVQ)

LVQ es una versión supervisada de los mapas auto-organizados de Kohonen (SOM) [Somervuo and Kohonen, 1999]. En dicho método se introduce cada uno de los ejemplos de entrenamiento en la red. Para cada ejemplo se calcula la célula ganadora de la capa F_2 de la misma forma que se hace con el método SOM. En la capa F_2 están los prototipos de las clases que producirá como salida la red. Así pues, la célula ganadora corresponderá al prototipo asignado al ejemplo de entrada, la clase a la que se le hará pertenecer. A continuación se realiza el aprendizaje para dicha célula ganadora, y las de su vecindario. Esto no puede hacerse en un aprendizaje supervisado, pues ya se conoce a priori a qué clase pertenece el ejemplo de entrenamiento. Así pues, se ha modificado el método para tener en cuenta esta información.

Las células de la capa F_2 constituyen los prototipos de la clasificación, y se distribuyen inicialmente de forma aleatoria por el espacio de estados. La solución al problema de clasificación será la colocación final de dichos prototipos, o células de la capa F_2, junto con la regla de vecinos más cercano. La colocación de los prototipos vendrá especificada por los valores de los pesos de las conexiones entre la capa de entrada y la capa de competición de la red. Mientras que la regla del vecino más cercano funciona de la siguiente manera:

- Se introduce el nuevo ejemplo a clasificar.
- Se calcula la distancia de este nuevo ejemplo a todos los prototipos existentes.
- Se etiqueta el nuevo ejemplo como perteneciente a la clase del prototipo cuya distancia calculada en el paso 2 sea más pequeña.

En un modelo de Red de Neuronas Artificial esta regla equivale simplemente a introducir un ejemplo en la red, propagarlo hasta obtener una salida y asignarle la clase a la que pertenece la célula ganadora.

Una vez distribuidos los prototipos de forma aleatoria, se van desplazando a medida que se van introduciendo los ejemplos de entrenamiento. Para ello se utiliza la misma regla de aprendizaje que el método de Kohonen para los SOM:

$$\frac{d\mu_{ij}}{dt} = \alpha(t)\tau_j(t)(\epsilon_i(t) - \mu_{ij}(t)) \tag{2.7}$$

Pero en este caso, la función $\tau_j(t)$ varía de la siguiente forma:

$$\tau_j = \begin{cases} 1 & \text{si } C_i \text{ ganadora y pertenece a la misma clase que } \epsilon_i \\ -1 & \text{si } C_i \text{ ganadora y no pertenece a la misma clase que } \epsilon_i \\ 0 & \text{en caso contrario} \end{cases}$$

Esta modificación de la función τ_j cambia sustancialmente el procedimiento con respecto al método de Kohonen. En primer lugar, al ser aprendizaje supervisado se ha eliminado el concepto de vecindario. Ahora sólo se modificará la célula ganadora, y las restantes no sufrirán ningún cambio. La modificación de la célula ganadora no es siempre en la misma dirección. En el método de Kohonen, la célula ganadora se aproximaba siempre al ejemplo introducido; de esta forma las células acababan en el centro de agrupaciones de ejemplos. En el método LVQ, si la célula ganadora pertenece a la misma clase que el ejemplo, la salida de la red es correcta, y para reforzarla se aproxima la red al ejemplo; de esta forma también se colocará cerca de los ejemplos a los que representa, y que son de su misma clase. Sin embargo, si la célula ganadora no pertenece a la misma clase que el ejemplo, se estará produciendo una salida incorrecta, y habrá que penalizar dicha salida. Para ello lo que se hace es alejar (de ahí el signo negativo de la función τ_j en el segundo supuesto) la célula del ejemplo para que en presentaciones posteriores de este mismo ejemplo haya otras células que lo representen, y no ésta. Así se consigue que los prototipos se vayan paulatinamente acercando a los ejemplos a cuya clase representan, y alejando de aquellos a los que no representan.

En este caso, el valor de la tasa de aprendizaje α se decrementa con el tiempo, al igual que ocurre con el método de Kohonen. Usualmente, este valor se inicializa con valores pequeños (0.1) y se decrementa de forma lineal muy ligeramente.

2.4.4. Predicción de Sistemas Dinámicos mediante Algoritmos Genéticos

Norman Packard[Meyer and Packard, 1992, Mitchell, 1996, Packard, 1990] desarrolló un sistema evolutivo basado en algoritmos genéticos y lo aplicó al campo del análisis de datos y predicción . Este problema se puede plantear de forma genérica de la siguiente forma: a partir de una serie de observaciones de un proceso, se anota un conjunto de pares:

$$\{(\bar{x}^1, y^1), ..., (\bar{x}^N, y^N)\}$$

donde $\bar{x}^i = (x_1^i, ..., x_n^i)$ son variables independientes, e y^i es una variable dependiente, con $(1 \leq i \leq N)$. Como ejemplo, en el caso de fenómenos climáticos, las variables dependedientes para cada día podrían ser la humedad ambiental, la presión barométrica media, la temperatura máxima y mínima, una variable booleana que indicase si llovió o no, y la variable dependiente podría ser si llueve o no al día siguiente. En un dominio de valores de acciones en bolsa, las variables dependientes podrían ser $\bar{x} = (x(t_1), x(t_2), ..., x(t_n))$ representando el valor de las acciones a lo largo de diferentes instantes de tiempo, y la variable dependiente $y = x(t_{n+k})$ sería el valor de dichas acciones en un instante de tiempo posterior (es decir, como una serie temporal).

Packard usó Algoritmos Genéticos para realizar búsquedas en el espacio del conjunto de condiciones sobre las variables dependientes, para encontrar aquellas que produjesen buenas predicciones para la variable dependiente. Por ejemplo, en el dominio de las acciones en bolsa, un invididuo de su GA podría ser una condición del tipo:

$$C = \{(\$20 \leq \text{Precio de las acciones de Xerox el día 1})$$

$$\wedge(\$25 \leq \text{Precio de las acciones de Xerox el día 2} \leq \$27)$$

$$\wedge(\$22 \leq \text{Precio de las acciones de Xerox el día 3} \leq \$25)\}$$

donde $\wedge$ representa al operador lógico "AND". Así pues, este individuo representa a todos los conjuntos de tres días en los cuales se cumplen las 3 condiciones (incluido el conjunto vacío si no se cumplen ninguna de las 3 condiciones). Por tanto, dicha condición C especifica un subconjunto dentro del conjunto de datos. El objetivo de Packard era usar GA para encontrar conjuntos de condiciones que fuesen buenos predictores. Es decir, buscar conjuntos de condiciones que definiesen subconjuntos dentro del espacio de datos cuyos valores de la variable dependiente estuviesen cerca de ser uniformes.

El fitness de cada individuo C se calculaba pasando todos los puntos $(\bar{x}, y)$ del conjunto de entrenamiento por C, buscando los $\bar{x}$ que cumpliesen la condición de C y guardando el correspondiente valor de y. Después, se usa una medida para calcular la uniformidad de los valores de y anotados. Si todos estos valores presentan poca varianza con respecto a su valor medio v, entonces la condición C es candidata a ser un buen predictor para y. Es decir, si un vector $\bar{x}$ cumple la condición de C, podemos esperar que su variable dependiente correspondiente y esté muy cerca del valor de v.

La función de fitness sugerida por Packard para una condición C fué la indicada en la ecuación 2.8.

$$f(C) = -log_2(\frac{\sigma}{\sigma_0}) - \frac{\alpha}{N_C} \tag{2.8}$$

donde σ es la desviación estándar del conjunto de y de los puntos del conjunto de entrenamiento que verificaban la condición de C, σ_0 es la desviación estándar de los valores de y para todos los datos del conjunto de entrenamiento, N_C es la cantidad de puntos que cumplieron la condición de C, y α una constante que se puede ajustar para cada dominio.

El esquema del Algoritmo Genético usado por Meyer y Packard era:

1. Inicializar la población con un conjunto aleatorio de Cs
2. Calcular el fitness de los Cs
3. Ordenar la población en base a su fitness.
4. Eliminar los peores individuos en base a un porcentaje de la población total, y reemplazarlos por los nuevos obtenidos mediante sobrecruzamiento y mutación.
5. Volver al paso 2.

El método de sobrecruzamineto elegido fué el uniforme. Aquí tenemos un ejemplo de su funcionamiento:

$$\text{Padre A} : \{(3{,}2 \leq x_6 \leq 5{,}5) \wedge (0{,}2 \leq x_8 \leq 4{,}8) \wedge (3{,}4 \leq x_9 \leq 9{,}9)$$

$$\text{Padre B} : \{(\mathbf{6{,}5 \leq x_2 \leq 6{,}8}) \wedge (\mathbf{1{,}4 \leq x_4 \leq 4{,}8}) \wedge (\mathbf{1{,}2 \leq x_9 \leq 1{,}7})$$
$$\wedge(\mathbf{4{,}8 \leq x_{16} \leq 5{,}1})$$

$$\text{Hijo A} : \{(3{,}2 \leq x_6 \leq 5{,}5) \wedge (\mathbf{1{,}4 \leq x_4 \leq 4{,}8}) \wedge (3{,}4 \leq x_9 \leq 9{,}9)$$

$$\text{Hijo B} : \{(\mathbf{6{,}5 \leq x_2 \leq 6{,}8}) \wedge (0{,}2 \leq x_8 \leq 4{,}8) \wedge (\mathbf{1{,}2 \leq x_9 \leq 1{,}7})$$
$$\wedge(\mathbf{4{,}8 \leq x_{16} \leq 5{,}1})$$

El este ejemplo, el Hijo A tiene dos genes del Padre A y dos genes del Padre B, minetras que el Hijo B tienen un gen del Padre A y tres genes del Padre B. Los operadores de mutación usados fueron:

- Añadir una nueva condición:

$$\{(3{,}2 \leq x_6 \leq 5{,}5) \wedge (0{,}2 \leq x_8 \leq 4{,}8)\}$$
$$\rightarrow \{(3{,}2 \leq x_6 \leq 5{,}5) \wedge (0{,}2 \leq x_8 \leq 4{,}8) \wedge (\mathbf{3{,}4 \leq x_9 \leq 9{,}9})\}$$

- Eliminar una condición:

$$\{(3{,}2 \leq x_6 \leq 5{,}5) \wedge (0{,}2 \leq x_8 \leq 4{,}8) \wedge (3{,}4 \leq x_9 \leq 9{,}9)\}$$
$$\rightarrow \{(3{,}2 \leq x_6 \leq 5{,}5) \wedge (0{,}2 \leq x_8 \leq 4{,}8)\}$$

- Acortar un rango:

$$\{(3{,}2 \leq x_6 \leq 5{,}5) \wedge (0{,}2 \leq x_8 \leq 4{,}8)\}$$
$$\rightarrow \{(\mathbf{3{,}9 \leq x_6 \leq 4{,}8}) \wedge (0{,}2 \leq x_8 \leq 4{,}8)\}$$

- Agrandar un rango:

$$\{(3{,}2 \leq x_6 \leq 5{,}5) \wedge (0{,}2 \leq x_8 \leq 4{,}8)\}$$
$$\rightarrow \{(3{,}2 \leq x_6 \leq 5{,}5) \wedge (\mathbf{1{,}2 \leq x_8 \leq 5{,}8})\}$$

2.4.5. Diseño Evolutivo de Clasificadores por Vecindario

El ENCC[Fernández and Isasi, 2004, Fernández and Isasi, 2008] (Evolutionary Design of Nearest Neighbour Classifiers) o diseño evolutivo de clasificadores por vecindario , es una aproximación evolutiva al problema del diseño de clasificadores por vecindario. Para este algoritmo no se requiere fijar el número de prototipos. Al igual que en los algoritmos propuestos en esta tesis, esta aproximación también sigue una perspectiva de Michigan, donde cada cromosoma representa un único prototipo, y no todo el sistema clasificador. Así pues, el sistema clasificador queda representado por toda la población. Los conceptos usados por este algoritmo son:

- Clasificador/Población, C: El conjunto de N prototipos $C = \{r_1, ...r_{,N}\}$
- Prototipo/Animal, r_i: Cada individuo se compone de la localización del mismo, y de la clase a la que pertenece.
- Región, r_i: El entorno esta dividido en un conjunto de N regiones. Cada individuo o animal se alimentará de los vegetales de su propia región. La región a la que pertenece cada animal viene definida por su posición y la regla del vecino más cercano.

- Clase/Especie, s_j: Tanto animales como vegetales pertenecen a una clase o especie del conjunto $S = \{s_1, ..., s_L\}$. El objetivo de un animal r_i de la especie s_j es comer tantos vegetales de la clase s_j como le sea posible, sin comer vegetales de otras clases s_k, con $k \neq j$.
- Patrón/Vegetal, v_r: Serán cada uno de los patrones o ejemplos usados para entrenar el sistema. Se consideran como vegetales por el algoritmo descrito en esta sección.
- Calidad/Salud de un individuo: Representa una relación ponderada entre el rendimiento local del prototipo y los rendimientos de los demás prototipos.

La segunda diferencia de este algoritmo con otros enfoques evolutivos está en los operadores usados para evolucionar la población. Durante el proceso de aprendizaje se aplican diversos operadores sobre cada individuo, y cada iteración se considera un año en la vida del animal. Este año se divide en 4 fases o temporadas: primavera, verano, otoño e invierno. En cada temporada, se aplicarán diferentes operadores, que aparecen descritos en la Tabla 2.1.

Tabla 2.1: Fases del Algoritmo ENCC

Temporada	Operador	Descripción
Primavera	Mutación	Cada animal cambia su especie por la de la mayoría de las especies de vegetales en su región
Verano	Reproducción	Los animales se reproducen para crear más animales que coman lo que los padres no quieren comer
Otoño	Luchar y moverse	Los animales luchan entre ellos y se mueven a una posición diferente para conseguir mas comida
Invierno	Muerte	Los animales más débiles mueren

Otro factor importante es que esta división en diferentes interaciones nos permite usar diferentes patrones de entrenamiento en cada una de las iteraciones. A continuación, se describen tanto la iniciañización del algortimo como cada una de las temporadas y operadores.

- Inicialización: Esta parte del algoritmo tiene dos posibles alternativas. La primera es empezar con un único individuo como población inicial.

La segunda alternativa es empezar con un individuo por cada patrón. Dependiendo del dominio, cada una puede obtener mejores resultados, pero en cualquier caso ambas son válidas.

- Primavera: Es la época en la que nacen los vegetales. Todos los animales son situados en su propia región, y recolectan los vegetales de su región. La forma de ver a qué animal pertenece cada vegetal se hace mediante la regla de vecino más cercano.

  Al final de la primavera, cada animal sabe la cantidad de vegetales de cada especie que pueden comer, con lo cual tenderá a la especie vegetal más abundante en su zona. Esta fase se corresponde con la fase de etiquetación en los sistemas tradicionales de aprendizaje supervidado. Para este algoritmo, el operador descrito se llama de mutación.

- Verano: Esta es la época en la que los animales se reproducen (segundo operador). En este caso, la reproducción es asexual, y un animal se reproduce solamente si necesita otro animal que se coma los vegetales que el no quiere. Este proceso es equivalente en el campo de las redes neuronales a introducir nuevas neuronas en la red, basándose en los aciertos de la misma.

  Así pues, un animal sólo se reproduce si hay más de un tipo de vegetales en su región. La probabilidad de reproducción es proporcional a la diferencia entre el número de clases vegetales en cada región. Los animales recién nacidos se situarán de forma que mejoren las aptitudes del padre.

- Otoño: Esta es la época en que la comida empieza a escasear, y los animales deciden buscar más comida. El otoño consta de dos fases. En la primera, los animales tienen la posibilidad de luchar entre ellos para arrebatarse los territorios. En la segunda fase, los animales se recolocan para encontrar el lugar óptimo en que pasar el invierno.

  - Luchar: Un animal puede decidir luchar contra otros para conseguir más comida. El operador de lucha se ejecuta para cada animal y consta de las siguientes fases:
    - Se elige un rival, asignando a cada posible rival una probabilidad proporcional a la distancia con el resto de los animales, usando una selección mediante ruleta.
    - Una vez elegido el rival, el animal debe decidir si enfrentarse a el o no. La probabilidad de luchar es proporcional a la diferencia de salud entre ambos rivales.

- Una vez decidido el rival, y si el animal decide luchar con el, hay dos posibilidades:
  - Si el animal al que se enfrenta no pertenece a su misma especie, no hay necesidad de confrontación, ya que no comparten alimento. Ambos llegan a un acuerdo para compartir región y que cada uno coma sus vegetales.
  - Si ambos animales pertenecen a la misma especia, ambos se enfrentan. La victoria será proporcional a la salud que cada uno tenga. El ganador toma la comida del perdedor.

- Moverse: El operador de movimiento permite recolocar cada animal en el mejor lugar posible para pasar el invierno, y esperar a la siguiente primavera. Así pues, cada animal se mueve al centro de los vegetales de su misma clase.

- Invierno: En esta época, los animales más débiles morirán. La probabilidad de morir es de 1 menos el doble de su salud. Así pues, los animales más saludables sobrevivirán con una probabilidad de 1, mientras que los más débiles, con una por debajo de 0,5. Al final de esta época, todos los vegetales desaparecen.

La función de salud de un individuo r_i se define mediante la ecuación 2.9:

$$salud(r_i) = min(1, aciertos(r_i) * aportacion(r_i)) \tag{2.9}$$

A su vez definimos:

$$aportacion(r_i) = \frac{||V_{ij}||}{\frac{expectativa(s_j)}{2}}$$

$$aciertos(r_i) = \frac{||V_{ij}||}{||R_i||}$$

donde R_i es conjunto de patrones localizados en la región r_i y $||V_{ij}||$ es el número de prototipos localizados en la región r_i que pertenecen a la misma clase que el prototipo r_i. También definimos $expectativa(s_j)$ como el número de patrones que un prototipo r_i de la clase s_j se espera que clasifique correctamente. Es decir:

$$expectativa(s_j) = \frac{||S_j||}{regiones(s_j)}$$

S_j se define como el conjunto de patrones que pertenecen a la clase s_j.

2.5. Objeciones a los Algoritmos Previos

La mayoría de los sistemas de aprendizaje automático encontrados sólo trabajan desde una perspectiva de aproximación global. Por contra, la mayoría de los sistemas capaces de trabajar a nivel local se quedan en labores de clasificación, y no llegan a trabajar con aproximaciones de funciones. Para este trabajo se ha considerado de más interés la tarea de aproximación que la de clasificación, tal y como se explicó en la sección 2.2.3.

Las primeras investigaciones de esta tesis se centraron en mejorar el sistema de Packard, explicado anteriormente. Las mejoras que se plantearon fueron:

- Usar sistemas de aproximación más potentes que una simple media. En el caso de esta tesis, una regresión lineal.
- Usar una perspectiva de Michigan para implementar el Algoritmo Genético.
- Usar una función de evaluación menos costosa que la de Packard.

Si bien la idea en un principio fué esta, y originalmente se planteaban otros modelos de aproximación embebidos dentro del sistema de Packard (por ejemplo redes neuronales), finalmente nos decantamos por desarrollar un sistema que corrigiese las carencias del modelo de Packard.

Capítulo 3

Objetivos de la Tesis Doctoral

Como se ha visto en el capítulo 2, la mayoría de los sistemas de aprendizaje automático realizan una labor de aprendizaje global, prestando poca o nula atención a los casos locales. Así mismo, este proceso de aprendizaje carece de un sistema de detección de casos extremos o impredecibles. Dado que la mayoría de los algoritmos de aprendizaje automático tienden a focalizar su aprendizaje en los ejemplos que más se repiten, en caso de recibir un patrón que describa un comportamiento anómalo, la tendencia de la mayoría de estos algoritmos será la de producir una salida estándar.

La idea básica de esta tesis consiste en el desarrollo de algoritmos que busquen subconjuntos de datos que presenten similitudes, al tiempo que se crean las funciones para predecir sobre estos subconjuntos.

El paradigma usado será el de los algoritmos evolutivos, que nos permitirá codificar y evolucionar individuos. Gracias a la versatilidad de estos algoritmos, dichos individuos podrán representar simultáneamente un subconjunto del espacio de datos, y una regla de predicción que se aplicará a los elementos de dicho subconjunto.

3.1. Esquema básico de los Algoritmos Desarrollados

A continuación se describe el esquema básico planteado en esta tesis para los algoritmos evolutivos. Para esta aproximación, lo ideal es usar una perspectiva de Michigan, tal y como vimos en el capítulo 2. En esta perspectiva, la solución al problema será toda la población, y cada individuo

representará una solución local.

3.1.1. Codificación de los Individuos

Todo individuo deberá estar compuesto de dos partes:

- una parte "condición" que represente un subconjunto de entrada.
- una "predicción" que devuelva un valor real para el punto que cumpla la regla.

Para aclarar lo anterior, veamos el siguiente ejemplo. Supongamos que trabajamos en dimensión n. Los patrones serán puntos de $\mathbb{R}^{n+1}$, ya que las n primeras coordenadas representan las variables de entrada, y la última la variable de salida. Un individuo I estará compuesto de dos pares. Es decir:

$$I = (C_I, P_I)$$

donde C_I es la condición asociada a I, y P_I su predicción asociada. Estas C_I y P_I serán dos aplicaciones:

$$C_I : \mathbb{R}^n \to \{0, 1\}$$

$$P_I : \mathbb{R}^n \to \mathbb{R}$$

Así pues, dado un punto $\bar{x} = (x_1, ..., x_n)$ del espacio de entrada, diremos que pertenece al subconjunto definido por C_I, o lo que es lo mismo, que cumple la condición de C_I, si y sólo si $C_I(\bar{x}) = 1$. Si en caso contrario $C_I(\bar{x}) = 0$, entonces diremos que $\bar{x}$ no pertenece al espacio definido por C_I, o equivalentemente, que no cumple la condición C_I. Para los puntos $\bar{x}$ que cumplan la condición C_I existirá una predicción, que será el valor que nos devuelva $P_I(\bar{x})$.

Para los algoritmos presentados en esta tesis, P_I será simplemente una regresión de la variable de salida sobre las variables de entrada para los patrones que cumpliesen la condición C_I.

Así pues, un individuo será representado como

$$I = (c_1, ..., c_k, p_1, ..., p_{n+1})$$

dónde $c_1, ..., c_k$ codificará la condición C_I, y $p_1, ..., p_{n+1}$ serán los $n + 1$ coeficientes de la regresión P_I.

3.1.2. Funcionamiento del Algoritmo Evolutivo

Dado un patrón $P = (p_1, ..., p_n, p_{n+1})$, suponiendo que cumpla la condición C_I, es decir, que:

$$C_I(p_1, ..., p_n) = 1$$

la predicción de I será mejor cuanto menor sea el error $e = |P_I(p_1, ..., p_n) - p_{n+1}|$. Así pues, es evidente que este error debe ser parte de función de evaluación del algoritmo evolutivo: tratando de minimizar este error, estamos haciendo más acertada su predicción. Con ello, además estamos haciendo que la condición C_I se ajuste más a los puntos que mejor prediga P_I. Supongamos que P es un patrón que cumple la condición C_I pero que P_I predice con una tasa de error demasiado alta. La función de evaluación perimitiría seleccionar un descendiente de I, que llamaremos $I' = (C'_{I'}, P'_{I'})$, siempre que mejorase a su progenitor. Esta posible mejora podría venir de dos formas:

- Alterando P_I de forma que $P'_{I'}$ permita predecir mejor el patrón P, o lo que es lo mismo, que su error de predicción sea menor:

$$|P'_{I'}(p_1, ..., p_n) - p_n| < |P_I(p_1, ..., p_n) - p_{n+1}|$$

- Alterando C_I de forma que P ya no cumpla la condición $C'_{I'}$.

En conclusión, la población de individuos representará un conjunto de reglas de predicción. Dado un patrón, la salida del sistema consistirá en buscar qué condición cumple dicho patrón, y a continuación devolver la predicción de dicha condición. En caso de que un patrón P no cumpliese ninguna condición, simplemente no habría predicción para este patrón, lo cual significaría que representa un comportamiento muy complejo e impredecible.

Es por tanto lógico, que los individuos tiendan a predecir lo mínimo posible. Al fin y al cabo, cuantas menos veces den una predicción, menos tenderán a equivocarse. Para evitar que el sistema tienda a generar individuos que no predigan nada, hay que premiar a los invididuos en función de cuantos patrones cumplen sus condiciones.

La selección de los individuos más aptos a lo largo del proceso de evolución permitirá que sus predicciones sean cada vez mejores, y con ello, que la predicción del sistema también sea más acertada.

3.1.3. Diferencias entre los algoritmos desarrollados en esta Tesis

El algoritmo esquematizado en esta sección describe por encima el funcionamiento de los algoritmos que se explicarán con detalle en los capítulos siguientes. Las diferencias entre ambos, para un individuo I:

- En el sistema de reglas, C_I representa una condición para las variables del patrón (o también se puede ver como la pertenencia a un hipercubo dentro del espacio de variables de entrada). Para el sistema de Voronoi, C_I representa una condición de pertenencia a una región de Voronoi, definida por un prototipo.
- La parte predictiva para dicho individuo (definida anteriormente como P_I) será común a ambos algoritmos: una regresión lineal.
- Mientras que la función de evaluación usada en el sistema de reglas será definida para cada individuo, en el sistema de Voronoi se usa una función de fitness global.

3.2. Objetivos

El principal objetivo de esta tesis será la búsqueda de algoritmos de aprendizaje basados en localización y estudio de las características locales de los datos. Estos algoritmos deberán automatizar las labores de búsqueda y predicción o aproximación. Para la realización automática y de forma inteligente, se usarán las herramientas del campo de la inteligencia artificial conocidas con el nombre de **Algoritmos Evolutivos**.

Como segundo objetivo, se requerirá que estos algoritmos tengan la capacidad de detectar los elementos extraños, o ruido, que no permitan generar conocimiento a partir de ellos. Esta capacidad será de gran importancia a la hora de predecir comportamientos anómalos e impredecibles.

Como tercer objetivo planteado en esta tesis, se ha buscado que los algoritmos desarrollados posean la capacidad de predecir con la precisión deseada.

Dado que también existen otros algoritmos capaces de realizar aprendizaje local, será obligatorio comparar los resultados de los algoritmos desarrollados con los previamente existentes.

Otra característica a exigir a los algoritmos desarrollados será la robustez, debiendo tener poca dependencia de parámetros e inicializaciones aleatorias.

3.3. Evaluación de la Tesis Doctoral

Para validar la consecución de los objetivos, se realizarán experimentos sobre varios dominios de diferente naturaleza. Los algoritmos investigados se probarán sobre distintas series temporales, con distintas características.

Para ello se ha buscado una serie artificial, la serie de Mackey-Glass y tres reales, el nivel de la marea en la laguna de Venecia, las manchas solares y el consumo de agua de un depósito. Dentro de las reales, la serie que mide el consumo de agua presenta un comportamiento estacionario, mientras que las otras, presentan un comportamiento cíclico. También y para probar la efectividad de los algoritmos, estos se aplicarán a un dominio bien distinto como el de la predicción en bolsa. En todos ellos se medirá alguna variante del error cuadrático medio sobre todos los patrones del conjunto de test (Error Cuadrático Medio, Raíz del ECM, o la normalización de ambos, según lo requiera el problema).

En el caso de las series temporales, además se realizarán varios experimentos aumentando el horizonte de predicción para analizar la evolución del error a lo largo de la línea temporal, y comprobar la resistencia de las capacidades predictivas de nuestros algoritmos a lo largo del tiempo.

Para contrastar los resultados, se harán comparativas de los mismos con los de otros sistemas de aprendizaje automático, implementados dentro del sistema WEKA (Waikato Environment for Knowledge Analysis - Entorno para Análisis del Conocimiento de la Universidad de Waikato). También se aplicarán test estadísticos para comprobar la significancia estadística de los resultados obtenidos por todos los sistemas. WEKA es la herramienta ideal para nuestros propositos, ya que no sólo implementa muchas de las herramientas más usadas en el área de aprendizaje automático (Perceptrón Multicapa, Regresiones Lineales, Árboles de decisión,... etc), sino que también incluye herramientas para el aprendizaje local (LWL, IBK, Redes Neuronales de Base radial,... etc).

También se comprobará mediante un test estadístico T de los resultados la dependecia de las inicializaciones aleatorias, así como la significancia estadística.

Capítulo 4

Sistema de Reglas

4.1. Introducción

En este capítulo analizaremos el primer sistema de predicción desarrollado en esta tesis. Dicho sistema trabaja sobre conjunto de datos que representan un dominio, el cual puede contener ruido. Para dar robustez al modelo debemos dotarlo de la capacidad de distinguir qué elementos del conjunto de datos pueden generar conocimiento y cuáles son inútiles o ruido. El sistema desarrollado en este trabajo está basado en algoritmos genéticos, y su objetivo es la búsqueda de reglas de predicción que detecten comportamientos locales en el conjunto de datos.

Este algoritmo de búsqueda automática de reglas está basado en los trabajos de Packard[Meyer and Packard, 1992, Mitchell, 1996, Packard, 1990] sobre predicción de sistemas dinámicos , usando algoritmos genéticos para generar reglas de predicción sobre una serie temporal. En este trabajo se han aplicado algunas técnicas avanzadas de computación evolutiva para alcanzar mejores resultados sobre problemas más generales que las series temporales.

4.2. Algoritmos evolutivos para la búsqueda de reglas de predicción

En algunos dominios, como ocurre en el caso de problemas de predicción de fenómenos naturales, la utilización de técnicas de aprendizaje automático se enfrenta con determinados problemas. Habitualmente, las técnicas de aprendizaje automático, y más concretamente los Algoritmos Genéticos, basan su aprendizaje en un conjunto de ejemplos. Si estos ejemplos están, fundamentalmente distribuidos a lo largo de determinados valores, será en

ese rango en el que se producirá el aprendizaje, y tenderán a tratar el resto de los valores como ruido. Esto que para algunos dominios es positivo, se convierte en un inconveniente en otros dominios. Por ejemplo en el caso de predicción en bolsa o predicción de mareas, por mencionar dos dominios muy diferentes, la mayoría de las medidas que existen están alrededor de los valores medios a corto plazo. En pocas ocasiones se producen grandes subidas o bajadas de la bolsa, o del nivel del agua. Sin embargo son precisamente esas situaciones las que tienen mayor importancia desde el punto de vista de la predicción. Este algoritmo se basa en la búsqueda de ambos tipos de comportamientos, los normales y los atípicos.

Para evitar el problema de la generalización se implementó una perspectiva de Michigan [Booker et al., 1989] en el algoritmo genético, usando una estrategia de Steady-State (o estado estacionario). En la perspectiva de Michigan, la solución al problema es la población total de individuos, en vez del individuo más apto, que es lo que ocurre en un algoritmo genético estándar. De esta forma se podría permitir la evolución de reglas para situaciones habituales del problema, pero también la evolución de reglas para situaciones críticas, evitando el que estas últimas situaciones sean consideradas como parte del problema, y predichas de forma incorrecta. Como cada regla se evalúa sólo con la parte de ejemplos que equiparan con ella, puede ser igualmente válida, aunque sea sólo parcialmente aplicable. Es precisamente esta característica de localidad la que hace que se pueda especificar en situaciones particulares. Por otra parte, este método no asegura que todos los ejemplos de pruebas vayan a poder ser predichos. Hay que intentar encontrar un balance entre el rendimiento del sistema, y su capacidad de predicción.

4.3. Codificación de las reglas

El objetivo de este trabajo es buscar reglas de predicción. En primer lugar debemos fijar un valor para D, que representa el número de variables de entrada para las reglas. Si se considera que la predicción depende, por ejemplo, de 5 variables ($D = 5$), una regla sería una condición del tipo "si el valor de la variable 1 es menor que 100 y mayor que 50, el de la variable 2 es menor que 90 y mayor que 40, el de la variable 3 es menor que 5 y mayor que -10 y el de la variable 5 es menor que 100 y mayor que 1, entonces el valor para la variable de salida será el valor de cierta función p_R con un error aproximado de 3". Esta regla podría expresarse de la siguiente forma:

$$\text{IF } (50 < x_1 < 100)\ AND\ (40 < x_2 < 90)$$

$$AND\ (-10 < x_3 < 5) AND\ (1 < x_5 < 100)$$

$$\text{THEN predicción} = p_R(x_1, x_2, x_3, x_4, x_5) \pm 3$$

Las reglas descritas de esta forma tendrían una representación gráfica como la de la figura 4.1. De forma genérica una regla (R) estaría compuesta por una parte condicional (C_R) y una parte predictiva (P_R). La parte condicional sería un conjunto de pares de intervalos del tipo $C_R = \{I_1^R, I_2^R, ..., I_D^R\}$; donde cada I_i^R es un intervalo $I_i^R = \{LI_i^R, LS_i^R\}$, con LI_i^R siendo el límite inferior y LS_i^R el límite superior respectivamente para la i-ésima variable de entrada. La parte predictiva estaría compuesta por dos valores, la predicción y el error, $P_R = \{p_R, e_R\}$. Para esta tesis, hemos usado como funciones predictivas regresiones lineales, pero en principio se podrían usar otras distintas. Por ejemplo, Packard [Packard, 1990] usa una función constante que devuelve un valor fijo.

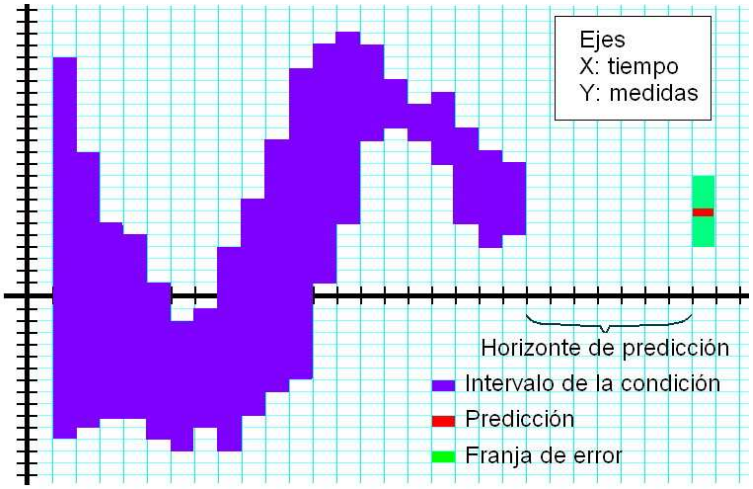

Figura 4.1:
Representación gráfica de una regla para un dominio de Serie Temporal

La parte condicional puede llevar valores comodín ($*$) que indican que el intervalo correspondiente es irrelevante, es decir, que no es necesario tenerlo en consideración. La regla anterior en forma de individuo sería: (50, 100, 40, 90, -10, 5, *, *, 1, 100, p_R, 5) . En la figura 4.2 se puede ver otra representación de un conjunto de 7 reglas recubriendo un espacio de entrada de 3 variables.

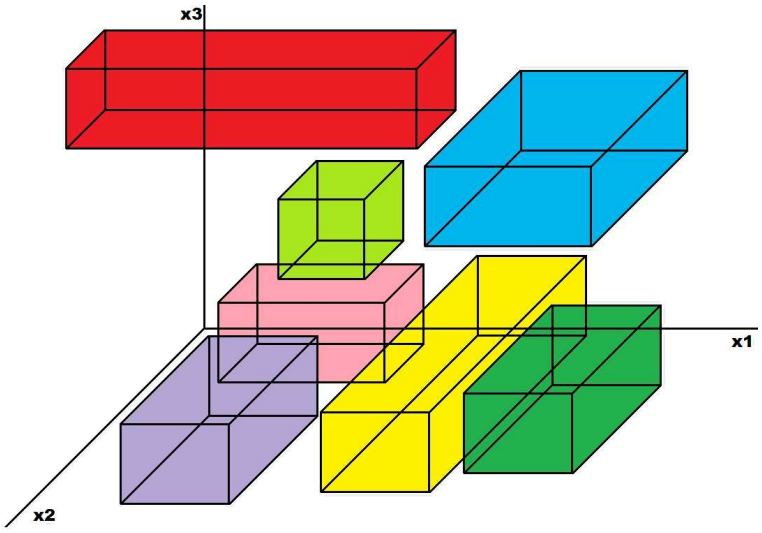

Figura 4.2:
Hipercubos recubriendo el espacio de entrada

De esta forma, se pueden codificar todas las reglas en individuos representados por vectores de tamaño $2D + (D + 1) + 1$, donde D es el número de variables de entrada del dominio. En un principio, usaremos $D + 1$ valores del individuo para codificar los coeficientes de la regresión lineal p_R. Si quisiésemos usar otra función para predecir, como por ejemplo una red neuronal, archivaríamos en el individuo los valores U correspondientes a los pesos de la red entrenada, y por tanto los individuos serían vectores de dimensión $2D + U + 1$. Una vez codificadas las reglas en un individuo, se pueden aplicar algoritmos genéticos para generar nuevas reglas a partir de éstas. Para ello, primero se seleccionan dos reglas para que produzcan una nueva regla descendiente. Este descendiente heredará sus genes de los de sus progenitores, siendo cada gen un intervalo I_j. Para cada variable de entrada i, el descendiente puede heredar dos genes (uno de cada progenitor) con la misma probabilidad. Esto tipo de herencia se conoce como sobrecruzamiento uniforme. Este descendiente no heredará las funciones de predicción ni el valor de error estimado. Veámoslo con un ejemplo:

Progenitor A:

$$(50, 100, 40, 90, -10, 5, *, *, 1, 100, p_A, 5)$$

Progenitor B:

$$(\mathbf{60}, \mathbf{90}, \mathbf{10}, \mathbf{20}, \mathbf{15}, \mathbf{30}, \mathbf{40}, \mathbf{45}, *, *, \mathbf{p_B}, \mathbf{8})$$

Descendiente:

$$(50, 100, \mathbf{10}, \mathbf{20}, -10, 5, \mathbf{40}, \mathbf{45}, *, *, p, e)$$

Como se puede apreciar, el descendiente carece de los valores para la función de predicción (p) y el error (e). En la figura 4.3 se puede ver un ejemplo del proceso de sobrecruzamiento representado gráficamente.

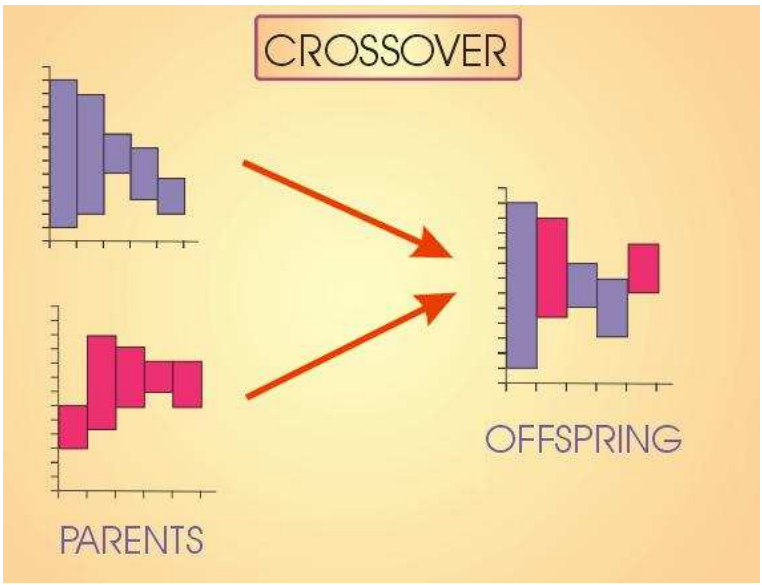

Figura 4.3:
Representación gráfica del sobrecruzamiento

Una vez engendrado el descendiente, hay una probabilidad del 10 % de que sufra mutación en alguno de sus genes. Este gen será seleccionado aleatoriamente. En la tabla 4.1 se muestran las mutaciones que puede sufrir el intervalo seleccionado (LI, LS). Todas estas transformaciones tienen la misma probabilidad de ser seleccionadas. $R(x, y)$ es una función que devuelve un valor aleatorio entre x e y, con $x < y$. El valor de W es un 10 % de la amplitud del intervalo definido por (LI, LS). Es decir, $W = 0{,}1(LI, LS)$. Obviamente, la transformación de (LI, LS) en (LI', LS') debe cumplir la condición $LI' < LS'$. En la figura 4.4 se muestran gráficamente estos procesos de mutación.

Tabla 4.1: Mutaciones

Nombre	Transformación
Nuevo valor aleatorio	$(LI, LS) \rightarrow (R(0,1), R(0,1))$
Condición nula	$(LI, LS) \rightarrow (*, *)$
Agrandar	$(LI, LS) \rightarrow (LI - R(0,1)W, LS + R(0,1)W)$
Reducir	$(LI, LS) \rightarrow (LI + R(0,1)W, LS - R(0,1)W)$
Mover arriba	$(LI, LS) \rightarrow (LI + cW, LS + cW)$
	$c = R(0,1)$
Mover abajo	$(LI, LS) \rightarrow (LI - cW, LS - cW)$
	$c = R(0,1)$

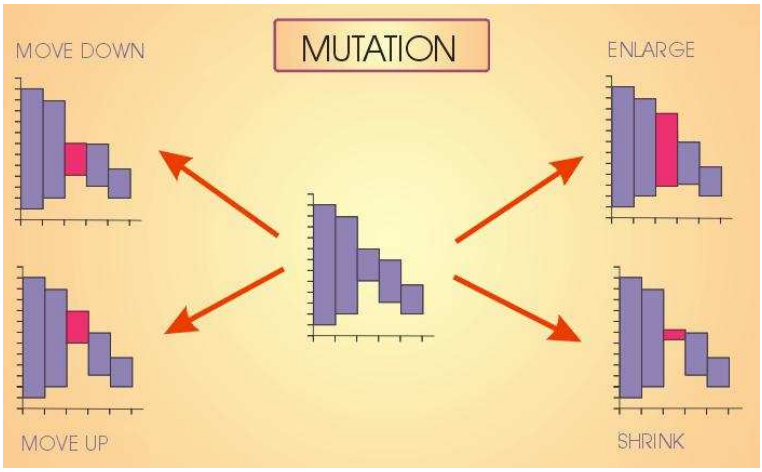

Figura 4.4:
Representación gráfica de los operadores de mutación

Los datos disponibles se repartirán en dos conjuntos, uno de entrenamiento y otro de validación, como es habitual en aprendizaje automático. Sea R un individuo, el proceso para calcular la predicción y el error de R usando el conjunto de entrenamiento será el siguiente:

- Dado el conjunto de patrones de entrenamiento S, se determina el subconjunto $C_R(S) \subset S$ de patrones $\vec{X}$ que cumplen la parte condicional de la regla R. Se llamará a este conjunto[1] $C_R(S)$:

$$C_R(S) = \{\vec{X} \in S | \vec{X} \; cumple \; C_R\}$$

[1]Es importante distinguir la condición C_R del conjunto $C_R(S)$.

donde $\vec{X} = \{x_1, x_2, ..., x_D, x_{D+1}\}$, siendo $x_1, x_2, ..., x_D$ las variables de entrada y x_{D+1} la variable de salida. El vector $\vec{X}$ cumple C_R si:

$$LI_1^R \leq x_1 \leq LS_1^R, LI_2^R \leq x_2 \leq LS_2^R, ...$$

$$..., LI_D^R \leq x_D \leq LS_D^R$$

- Se calcula la predicción p_R de la regla R realizando una regresión lineal con todos los vectores existentes en el conjunto $C_R(S)$. Para ello se determinarán los coeficientes de regresión $\vec{A} = \{a_0, a_1, ..., a_D\}$ que establecen el hiperplano que mejor aproxima en conjunto de puntos $C_R(S)$. Sea $p_R(\vec{Y})$ el valor estimado por la regresión para el patrón $\vec{Y} = (y_1, ..., y_D, y_{D+1})$, éste se calcula de la siguiente forma:

$$p_R(\vec{Y}) = a_0 y_1 + a_1 y_2 + \ldots + a_{D-1} y_D + a_D$$

- El error estimado de la regla, e_R, será:

$$e_R = Max\{|y_{D+1} - p_R(\vec{Y})| / \vec{Y} \in C_R(S)\}$$

De esta forma, cada individuo es una regla capaz de predecir parcialmente la serie. El conjunto de todos los individuos (todas las reglas) define el sistema de predicción. Sin embargo, puede ocurrir que haya patrones que no cumplan ninguna regla. En este caso se dice que el sistema no puede tomar una decisión para dichos patrones. Es deseable, y es un objetivo de este trabajo, que la cantidad de patrones sin predicción sea lo más pequeña posible. Nuestro sistema, por lo tanto, debe buscar individuos que, sobre el conjunto de entrenamiento, predigan el máximo numero de patrones con el mínimo error posible. Para ello, necesitamos definir una constante del algoritmo que llamaremos EMAX y cuyo objetivo es penalizar a los individuos con un error máximo absoluto mayor que su valor. Dicho valor será definido por el usuario en función de la exactitud de las predicciones que desee obtener del sistema. Así pues, definimos la siguiente función de $fitness$ para un individuo R (cuyo error es e):

```
IF ((NR>1) AND (e < EMAX)) THEN
      fitness = (NR*EMAX) - e
ELSE
   fitness = f_min
```

donde NR es el número de patrones del conjunto de entrenamiento (S) que satisfacen la condición C_R (es decir, NR es el cardinal del conjunto $C_R(S)$). Con la condición ($NR > 1$) nos aseguramos de que dicha condición fuese cumplida por más de un patrón durante la fase de entrenamiento. f_min es un valor mínimo que se asigna a los individuos cuya regla no se cumple para ningún patrón.

La función de fitness tiene como objetivo establecer un balance entre individuos cuyo error de predicción sea el más bajo posible, pero que al mismo tiempo, el número de patrones para los cuales hace predicción (la cardinalidad de $C_R(S)$) sea el más alto posible.

4.4. Inicialización

El método, así diseñado, tiende a mantener la diversidad de los individuos de la población en cuanto a su capacidad para predecir zonas diferentes del espacio de búsqueda. Sin embargo para mantener la diversidad, ésta tiene que existir previamente. Para conseguir esto se ha realizado un procedimiento particular de inicialización de la población, de forma que esta se distribuya de manera uniforme a lo largo del rango de entrada posible de los datos. Por ejemplo, en el caso de la predicción de mareas, el rango de entrada va desde los -50 cm, hasta los 150 cm. Si la población consta de 100 individuos, se pueden realizar 100 intervalos de 2 cm de anchura, y de esta forma abarcar todos los posibles valores de entrada. El procedimiento de inicialización crea una regla para cada uno de los intervalos antes mencionados, de forma que su parte derecha (la predicción) esté dentro del intervalo. Esto se realiza de la siguiente manera:

1. Definimos como $\mathbb{E}$ el conjunto de entrenamiento. El patrón $P^i \in \mathbb{E}$ tendrá n variables de entrada P_j, con $0 < j \leq n$ y una variable de salida P_s.

2. Se determinan todos los patrones del conjunto de entrenamiento $\mathbb{E}$ cuyas salidas estén dentro del intervalo considerado. Si I es el intervalo dado, y P_s la salida del patrón P, el conjunto $S_I \subset \mathbb{E}$ se define como:

$$S_I = \{P \in \mathbb{E} \mid P_s \in I\}$$

3. Para cada una de las n variables de entrada de los patrones se determina el mínimo valor máximo y el máximo valor mínimo de forma que

todos los patrones calculados en el paso anterior cumplan esos límites. Es decir, para cada i buscamos

$$MAX_j^I = Min\{x \mid x > P_j,\ \forall P \in S_I\}$$

$$MIN_j^I = Max\{x \mid x < P_j,\ \forall P \in S_I\}$$

4. Dichos intervalos serán los que posea la regla a generar, y su predicción será la media de las salidas de los patrones pertenecientes a S_I, y su error el máximo de los errores sobre la media. En otras palabras, la regla generada será R_I:

   $$R_I = (MIN_1^I, MAX_1^I, MIN_2^I, MAX_2^I, \ldots, MIN_n^I, MAX_n^I, p_{R_I}, e_{R_I})$$

   donde

   $$p_{R_I} = Media\{P_s \mid P \in S_I\}$$

   y

   $$e_{R_I} = Max\{|P_s - p_{R_I}| \mid P \in S_I\}$$

Este procedimiento producirá reglas muy generales, de forma que entre todas cubren la totalidad del espacio de búsqueda. El problema de las reglas obtenidas de esta forma es que habitualmente producen errores de predicción grandes. Será el Algoritmo Genético el que se encargará de depurar las soluciones y generar reglas más específicas y acertadas.

4.5. Evolución de las reglas

Definimos como fenotipo de un individuo, su zona de predicción, o lo que es lo mismo, en qué rango de la variable de salida se ha especializado en predecir. Para ello, para cada individuo calculamos la media de todas las predicciónes que realizó con los patrones de entrenamiento, y anotamos este valor como su fenotipo.

Como se ha mencionado, el método aportado en este trabajo consiste en realizar una perspectiva Michigan junto con un estrategia de estado estacionario. Esto quiere decir que en cada generación se seleccionan dos individuos, llamados progenitores, mediante selección proporcional a la función de evaluación. Esta selección se realiza mediante torneos de tres rondas. Los progenitores son cruzados para producir un individuo descendiente.

A continuación, se selecciona de la población aquel individuo, cuya distancia fenotípica con el individuo recién creado sea menor, es decir, se busca

aquel individuo en la población que realizara predicciones sobre zonas parecidas del espacio de búsqueda. El nuevo individuo sustituye al seleccionado de la población, sólo si su fitness es mayor, es decir sólo si es una mejor solución al problema. En caso contrario la población permanece inalterada. Podemos ver un ejemplo gráfico en la figura 4.5.

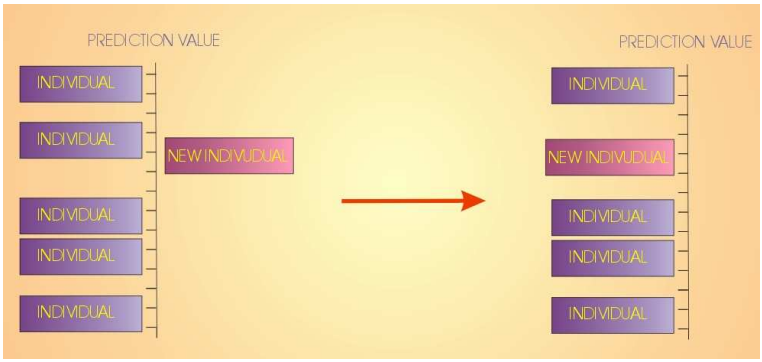

Figura 4.5:
Inserción de nuevos individuos

Esta forma de realizar la sustitución es típica de los métodos de crowding [Jong, 1975], en los que se intenta mantener la diversidad de la población para encontrar varias soluciones válidas. En este caso está aún más justificado, ya que la diversidad de la población es la que permitirá que se cubra la mayor parte del espacio de búsqueda, y que las reglas generadas puedan predecir un mayor número de situaciones. Además, la variabilidad y la conservación de la adyacencia de las soluciones permite que se generen reglas para situaciones especiales (grandes subidas de la marea por ejemplo) que de otra forma no habrían podido aparecer. El pseudocódigo de todo este proceso puede leerse en el algoritmo 2.

4.6. Predicción

Al ser un método estocástico, se obtienen distintas soluciones en diferentes ejecuciones. Por esto, tras cada ejecución se guardan las soluciones obtenidas, y al final del proceso, se unen a las obtenidas en el resto de las ejecuciones. El número de ejecuciones se determina en función del porcentaje del espacio de búsqueda cubierto por las reglas. El conjunto de todas las reglas obtenidas en las diferentes ejecuciones determina la solución final del sistema. Una vez obtenida la solución, es empleada para producir salidas

ante entradas desconocidas. Esto se realiza de la siguiente forma:

- Para cada nueva entrada se determina qué reglas de entre las de la solución obtenida se cumplen.
- Se obtiene una salida por cada una de las reglas que se han cumplido.
- Se realiza la media de las salidas de las reglas que se cumplieron, y el valor resultante es la predicción final producida por el sistema.

Tenemos un ejemplo de este procedimiento en la figura 4.6. En el ejemplo, se recibe un patrón desconocido del conjunto de validación, y las reglas que se cumplen son la 1,2,3,5,6,8,9 y 10. Cada una de las reglas cumplidas produce una predicción, con lo cual tenemos 8 predicciones. La salida final del sistema es la media de estas 8 predicciones. Si se diese el caso de que ninguna de las 10 reglas hubiese producido predicción, entonces el sistema no produce predicción final.

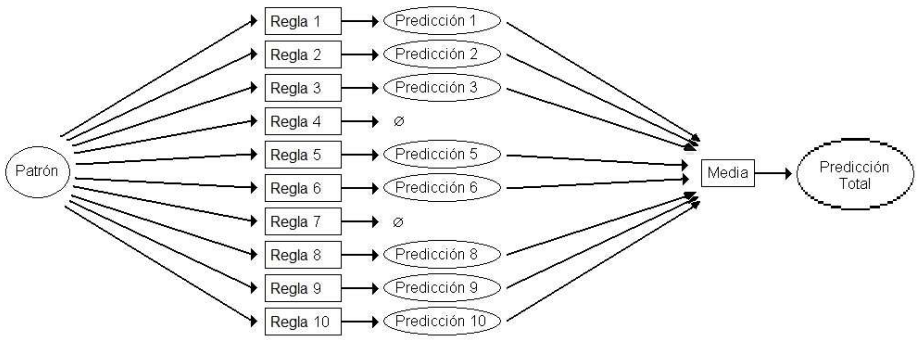

Figura 4.6:
Esquema del proceso de predicción

Para dominios discretos podría sustituirse la media por la moda, habilitando un procedimiento para decidir los empates.

En algún momento del desarrollo de este sistema, se planteó que se trabajase con reglas condicionales mas complejas, que incluyesen no sólo AND, sino también OR. Un ejemplo sería

$$\text{IF } (50 < x_1 < 100)\ AND\ ((40 < x_2 < 90)\ OR\ \ (-1 < x_2 < 0))$$

$$AND\ ((-10 < x_3 < 5)\ OR\ (1 < x_3 < 4))\ AND\ (1 < x_5 < 100)$$

THEN predicción $= 33 \pm 3$

Pero como sabemos por la lógica de predicados, cualquier sentencia se puede expresar de la forma $a_1 \vee a_2 \vee ... \vee a_n$, donde cada a_i es del tipo $a_i = b_1^i \wedge b_2^i \wedge ... \wedge b_m^i$, y cada b_j^i es atómica (en nuestro caso, b_j^i es del tipo $(z < x < y)$). Así pues, dado que para nuestro sistema, semánticamente es lo mismo tener una regla $a \vee b \rightarrow c$ que dos reglas $a \rightarrow c$ y $b \rightarrow c$ por separado, nos decantamos por mantener la gramática más simple, ya que hacía más simples los invididuos, y más rápidos de evaluar.

Algorithm 2 Algoritmo para la búsqueda de reglas locales

Pseudocódigo para la búsqueda de reglas locales
$R = \Theta$

while (Predicción del nivel de $R < \theta$) **do**
P = población inicial de reglas
Calcular la predicción $p_i \in P$ mediante regresión lineal.
Archivar la zona de predicción z_{p_i} como la media de todas las predicciones del individuo p_i
Evaluar cada $p_i \in P$. Sea e_{p_i} la evaluación del inidviduo i.
$generation = 0; stopCondition =$ FALSE;

while ($generation < maxGenerations$) AND (NOT $stopCondition$) **do**
Seleccionar dos individuos (p_1 y p_2) mediante torneos
Generar un descendiente (o) sobrecruzando los individuos seleccionados
o = Mutar (o,probabilidad)
Generar la predicción de o mediante regresión lineal.
z_o = Zona de predicción de o
e_o = Evaluación de o
Encontrar el individuo $q \in P$ cuya zona de predicción z_q sea la más cercana a z_o
if ($e_q < e_o$) **then**
Reemplazar q por o
$generation = generation + 1;$

end while
$R = R + P$

end while

Capítulo 5

Sistema de Voronoi

En este capítulo, describimos un nuevo sistema [Luque et al., 2009] de aprendizaje supervisado basado en estrategias evolutivas [Bäck et al., 1991] aplicado a la predicción de series temporales. Generalmente, esta clase de sistemas usa el proceso de entrenamiento para encontrar un modelo que nos permita predecir futuros comportamientos de la serie temporal. Sin embargo, en algunos dominios, las particularidades de la serie conceden gran importancia a los comportamientos locales. Para muchas series suele haber diferencias significativas entre las diversas regiones del espacio de la entrada. El sistema presentado en este capítulo busca automáticamente las áreas en el espacio de la entrada que comparten características predictivas, a la vez que ajusta dichas predicciones. Así, a la vez que va dividiendo el espacio de la entrada en diversas áreas, por medio de un sistema de prototipos y de vecino más cercano, va produciendo un método predictivo para cada región.

Como en la mayoría de los métodos de aprendizaje automático, se requieren dos fases: una de entrenamiento para ajustar el sistema y crear un modelo, y otra de validación, para comprobar que el aprendizaje ha sido correcto. Para nuestro algoritmo, la fase de entrenamiento constará de una primera parte en la cual se divide el espacio de entrada en regiones de Voronoi. A continuación, para cada una de estas regiones, se calculará una regresión lineal a partir de los patrones pertenecientes a dicha region. Posteriormente, y en diferentes iteraciones, deberemos ajustar estas regiones de forma que las predicciones de las regresiones asociadas a cada región sean lo más precisas posible. Este último paso del algoritmo se hará mediante estrategias evolutivas.

Una vez finalizado el proceso de entrenamiento, el sistema deberá ser capaz de producir una salida para cada patrón del conjunto de validación.

Esta salida se obtiene mediante la región de Voronoi a la que pertenezca este patrón de validación. Una vez asignado este patrón a una región, la salida del sistema será la predicción de la regresión asociada a dicha región para dicho patrón. Un aspecto interesante de este algoritmo es su capacidad para localizar los patrones que representan ruido en la muestra en regiones específicas.

5.1. Mejorando el Sistema de Reglas

Una vez que ya se había desarrollado el sistema de reglas, se plantearon diversos caminos para su mejora. Finalmente, se optó por el que nos pareció más interesante, que fué buscar un método que se adaptase mejor al conjunto de datos de entrenamiento. Si el espacio de datos es un subconjunto de $\mathbb{R}^{n+1}$ para algún $n \in \mathbb{N}$, una regla $(LI_1, LS_1, LI_2, LS_2, ..., LI_n, LS_n)$ será cumplida por los puntos de $\mathbb{R}^n$ que estén dentro del hipercubo definido por esta regla. Es decir, los puntos $x = (x_1, x_2, ..., x_n)$ tales que:

$$LI_i < x_i < LS_i, \text{ para todo } i \text{ tal que } 0 < i \leq n$$

Así pues, la forma que tiene el sistema de reglas de cubrir el espacio de entrada es mediante hipercubos que se pueden superponer o no. El caso es que el sistema no tiene por qué producir hipercubos para cubrir todo el espacio $\mathbb{R}^n$. Un patrón que represente un punto de dicho espacio que no sea cubierto por ninguno de estos hipercubos (o reglas) será un representante de un comportamiento atípico, y por tanto no producirá salida del sistema.

La idea del sistema que se detalla en este capítulo consiste en buscar una forma inteligente de cubrir todo el espacio de entrada, de forma no supervisada y manteniendo las características locales de los datos de entrada.

Para la división del espacio de entrada, se buscó un método análogo al que usan las Redes de Base radial o el Learning Vector Quantization [Somervuo and Kohonen, 1999] basados en regiones de Voronoi. La idea es que ahora cualquier patrón pertenecería a una región dentro del espacio de entrada, incluso aquellos patrones impredecibles o representantes de ruido.

A continuación se buscó un método para ajustar dichas regiones de Voronoi de la mejor forma posible a los datos, tal que minimizase el error de predicción.

5.2. Entrenamiento

La fase de entrenamiento se divide en varios procedimientos, siendo uno de ellos la división del espacio en regiones de Voronoi. Sea n la dimensión del espacio de entrada, la dimensión de los patrones será $n+1$. Definiremos como 'prototipo' a un punto P en el espacio de las variables de entrada (que será un subconjunto de $\mathbb{R}^n$). Dado un conjunto de k prototipos $\{P_i \in \mathbb{R}^n$, con $i \leq k\}$ el conjunto definido divide el espacio $\mathbb{R}^n$ en k regiones, como resultado de aplicar la regla del prototipo más cercano, es decir, un punto pertenece a la región definida por un prototipo si dicho punto está más cerca de éste que de los demás prototipos. Si definimos V_i como la región de Voronoi definida por P_i, dicha región se puede definir matemáticamente como:

$$Vi := \{P \in \mathbb{R}^n / d(P, P_i) < d(P, P_j) \text{ para todo } j \leq k \text{ siendo } j \neq i\} \quad (5.1)$$

donde d es la distancia euclídea.

El objetivo de esta fase es determinar la mejor partición del espacio de entrada a la hora de realizar las predicciones. Con este objetivo, se ha implementado una estrategia evolutiva en la cual cada individuo representa un prototipo. La evolución de los individuos consistirá en mover los prototipos que codifican de forma que mejoren la capacidad de predicción, en función de los patrones que pertenezcan a cada región.

5.2.1. Codificación

Para que los individuos evolucionasen, nos decantamos por el uso de las Estrategias Evolutivas debido a que proporcionan un mayor rendimiento en problemas que tratan con variables reales. En nuestro caso, las variables reales que debemos ajustar representan las coordenadas de los prototipos representados en cada individuo. A diferencia de los Algoritmos Genéticos estándar, en EE necesitamos un parámetro adicional para cada individuo, que representa la varianza de la mutación. Esta varianza nos indica como de cerca está este individuo de ser una buena solución: una varianza pequeña para este individuo significa que está bastante cerca de una solución del problema, mientras que una varianza grande nos indica que dicho individuo no se encuentra cerca de ninguna solución potencial, y por tanto necesita grandes cambios en sus coordenadas.[1]

[1] De hecho, aunque la bibliografía [Bäck et al., 1991, Bäck and Schwefel, 1992] sugiere el uso de una variable de varianza por cada coordenada del individuo, en trabajos previos

El esquema final del cromosoma de un individuo I podría expresarse como:

$$I = (x_1, ..., x_n, \sigma_I)$$

donde $x_1, ..., x_n$ son las coordenadas del prototipo que representa el individuo I (al cual llamamos P_I), y σ_I será la varianza de dicho individuo.

Tal y como se explicó anteriormente, para cada individuo I, el prototipo P_I define una región de Voronoi V_I (equacion 5.1). A continuación, se calcula una regresión lineal R_I a partir de los patrones que se encuentren en la región V_I.

5.2.2. Evaluación del Fitness

Los patrones de entrenamiento son vectores de $(n+1)$ coordenadas (una coordenada más que la dimensión del espacio de las variables de entrada). Para codificar los individuos tan sólo usaremos las n primeras coordenadas.

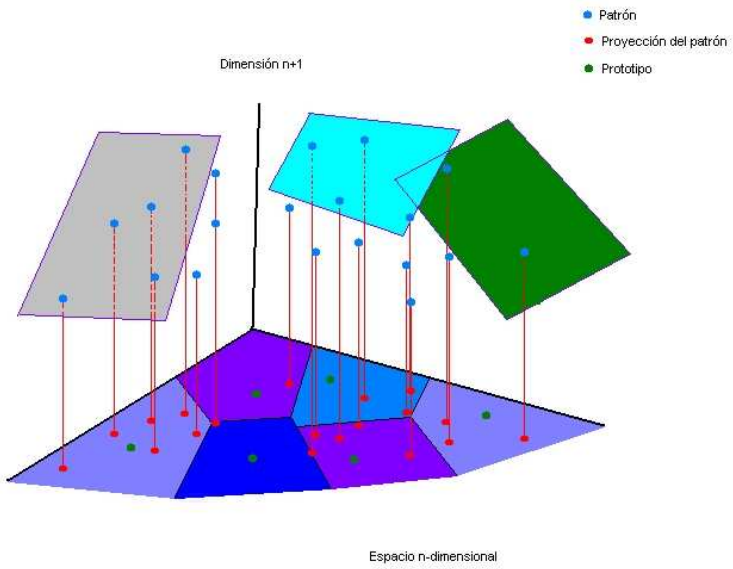

Figura 5.1:
Representación gráfica

Sea $T = (t_1, ..., t_{n-1}, t_n, t_{n+1})$ un patrón de entrenamiento, definimos la

[Luque et al., 2004a] se comprobó que es suficiente con una única varianza para todas las coordenadas, con las consiguientes ventajas para el algoritmo.

aplicación de proyección de $\mathbb{R}^{n+1}$ en $\mathbb{R}^n$ como:

$$\Pi(t_1, ..., t_{n-1}, t_n, t_{n+1}) := (t_1, ..., t_{n-1}, t_n)$$

A la hora de calcular el fitness de un individuo (o prototipo) P_i debemos asignar los patrones de entrenamiento a cada prototipo. Esto se hará mediante la regla del vecino más cercano. Dado un patrón T:

$$\Pi(T) \in V_i \iff d(T, P_i) < d(T, P_j) \text{ para todo } j \leq k \text{ siendo } j \neq i$$

Cuando ya se han asignado todos los patrones de entrenamiento a su correspondiente región, se procede a calcular la regresión asociada a cada región. Definimos R_i como la regresión lineal de la variable de salida sobre las variables de entrada de los patrones T tales que $\Pi(T) \in V_i$. Sea $R_i(T)$ la estimación de la regresión R_i para la salida del patrón T, y sea T_O la variable de salida del patrón T, es decir, si $T = (t_1, ..., t_{n-1}, t_n, t_{n+1})$, entonces $Out(T) := t_{n+1}$. Así pues, el error de la estimación será $E_T = |Out(T) - R_i(T)|$. Por tanto, tenemos una relación $P_i \rightarrow V_i \rightarrow R_i$ para cada $i \leq k$. Esta relación significa que cada prototipo P_i define una región V_i, la cual tiene asociada una regresión R_i. Todo esto se puede apreciar en la figura 5.1.

El objetivo del método presentado es minimizar el sumatorio de los errores. Los algoritmos evolutivos estándar asignan una puntuación de fitness a cada individuo. Siguiendo este enfoque, si hubiésemos usado como función de fitness para un individuo la suma de los errores de la estimación para cada patrón perteneciente a la región definida por dicho individuo, nos encontraríamos con el problema de que los individuos con menos patrones en su región obtendrían un fitness muy bajo (y que por tanto sería mejor dado que nuestro objetivo es minimizarlo). Si en vez de esta función de fitness se usase el error medio de las estimaciones, tendríamos el problema de que los individuos con muchos patrones en su región tenderían a evolucionar tomando patrones de los individuos limítrofes, empeorando inapreciablemente su fitness (ya que añadir un patrón a su región apenas modificará la media de errores), y a su vez, empeorando visiblemente el fitness de los individuos robados.

En nuestro sistema hemos optado por era usar un fitness global. Esto es, en vez de tener un valor de fitness para cada individuo, se usará una función que nos devuelva un valor para la toda población. El valor elegido fue la suma de todos los errores $E_T = |Out(T) - R_i(T)|$, para cada patrón T e i tal que $\Pi(T) \in V_i$. Matemáticamente:

$$\text{Fitness} := \sum_{T} |Out(T) - R_i(T)|,\ i \text{ tal que } \Pi(T) \in V_i$$

5.2.3. Evolución

La población inicial se crea aleatoriamente. A partir de ahí, el proceso de evolución funciona básicamente como una estrategia evolutiva $(1 + 1)$ paralela. Esto quiere decir que, en cada generación, cada individuo produce un descendiente por mutación de las coordenadas del padre. Sea $I = (x_1, ..., x_n, \sigma_I)$ un individuo, y sea $I' = (x'_1, ..., x'_n, \sigma_{I'})$ su descendiente, el proceso de mutación de podría expresar como:

$$x'_i = N(x_i, \sigma_I)$$
$$\sigma_{I'} = \sigma_I e^{\alpha N(0,1)}$$

Donde $N(X, Y)$ es una variable aleatoria normal de media X y varianza Y y $\alpha = 0{,}7$ una constante. Así mismo, σ_I y $\sigma_{I'}$ representan los valores de las varianzas del padre y el hijos respectivamente. Ambas ecuaciones para las mutaciones han sido extraídas de [Bäck et al., 1991].

Para cada nuevo individuo I' buscamos en la población el individuo $\hat{I}$ más cercano en términos de distancia euclídea. A continuación comparamos cuál de estos dos individuos es mejor para la población. Así pues, calculamos el fitness global, primero con $\hat{I}$ y posteriormente sustituyéndolo por I', y mantenemos aquel de los dos que produzca un mejor fitness global, descartando el otro. Este modelo de reemplazamiento tiene dos funciones:

- En un primer momento, cuando la varianza es muy grande para todos los individuos, obtendremos una fase de exploración de las soluciones. Esto quiere decir que gracias a unas tasas de mutación elevadas, los desdendientes de un individuo no tienen por qué estar cerca de su padre. Así pues, en un principio, un buen individuo no competirá con sus descendientes por los puntos de una zona del espacio, sino que mandará a sus hijos a competir con otros individuos. Así pues, estos hijos servirán para particionar regiones que lo necesiten.

- A lo largo de las iteraciones, la varianza de los individuos tenderá a decrementarse, transformando poco a poco la labor de exploración en explotación de soluciones. Mutaciones pequeñas generarán individuos cerca de sus progenitores, con los cuales competirán para mejorar la localización de los prototipos que representan.

Todo el proceso de entrenamiento se describe esquemáticamente en el algoritmo 3[2].

5.3. Predicción

Una vez creado el modelo durante el proceso de entrenamiento, procedemos a detallar el proceso de predicción. Para aumentar la robustez del sistema y hacerlo menos dependiente del azar (o de las inicializaciones), dada la naturaleza estocástica del algoritmo, se ha estructurado el sistema en dos capas. El proceso de predicción en la primera capa consiste en localizar para un patrón T a qué región de las codificadas por los individuos pertenece. A continuación, aplicamos a T la regresión asociada a dicha región para obtener la salida del sistema para dicho patrón.

El algoritmo descrito en este capítulo nos permite asignar cierta fiabilidad a las predicciones que produce. Ya que cuantos más puntos tenga una región, más puntos tendremos para calcular la regresión asociada, y por tanto más fiable será el resultado de dicha regresión. También podemos usar un valor mínimo de puntos, por debajo del cual, la predicción de la regresión asociada a una región no se considerará fiable. En tal caso, si se fuese a usar esta región para predecir el valor de un patrón T, en lugar de eso diremos que el sistema no produce una salida fiable. Todo el proceso de predicción del sistema para el patrón T se podría resumir en algoritmo 4, donde $\sharp R_i$ es el número de puntos que tenía la región R_i tras el proceso de entrenamiento, y MIN es la constante que nos determina el mínimo de puntos requeridos para que la predicción de considere fiable. A mayor valor de esta constante, más fiable será la predicción.

La segunda capa de predicción consiste en un conjunto de k sistemas predictivos o modelos entrenados por separado. Este parámentro k se decide en función de la seguridad que queramos para el sistema: cuanto más grande sea su valor, más fiable será la predicción de la segunda capa. Para los experimentos realizados, se ha comprobado que un valor $k = 10$ es suficientemente bueno.

Para conocer la salida producida por la segunda capa para un patrón dado, se manda dicho patrón a cada uno de los k sistemas entrenados por

[2]En realidad, no todas las regresiones deben ser recalculadas para la evaluación de la población total para cada nuevo individuo. Dado que las regresiones lineales tienen un alto coste computacional, el hecho de almacenar los resultados de las regiones de Voronoi que no sufren alteraciones al introducir un nuevo individuo en la población supone un gran ahorro de tiempo de cómputo. La versión del pseudocódigo mostrada aquí, es una versión simplificada, que no tiene en cuenta esta optimización.

separado, y cada uno dará una salida o ninguna. La salida final de la segunda capa del sistema será la media de los valores devueltos por los k sistemas entrenados por separado, o incluso puede que no se produzca ninguna respuesta, si ninguno de los sistemas devolvió respuesta alguna.

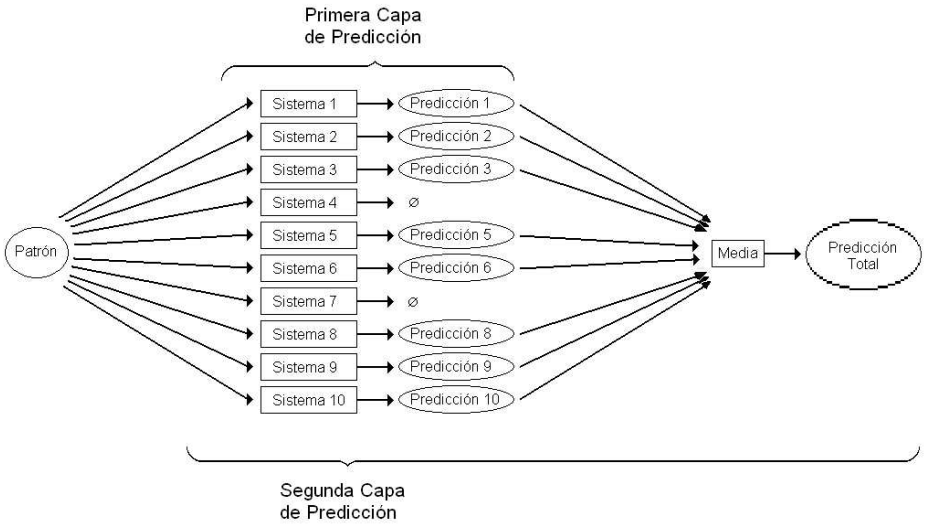

Figura 5.2:
Esquema de la predicción en 2 capas

En la figura 5.2 se muestra un esquema del funcionamiento de la predicción en 2 capas.

5.4. Ventajas del sistema

Destaca el hecho de que este sistema, y al igual que ocurría con el Sistema de Reglas, puede trabajar con patrones genéricos, y no sólo con los obtenidos a partir de una serie temporal. Así pues, se puede aplicar a cualquier dominio de regresión que se pueda representar con patrones.

La ventaja que aporta este sistema al anterior es su mayor adaptabilidad a los datos del espacio de entrada: este se puede recubrir mejor con regiones de Voronoi que con hipercubos. Mientras que el SRR cubría sólo las regiones que contenían información interesante, el SV lo recubre todo, y a la hora de predecir se descartan las regiones que no contenían información interesante, esto es, información representativa y que no esté alterada por ruido.

Las ventajas sobre los algoritmos de aprendizaje estándar que este sistema comparte con el anterior son:

- Media de diferentes predicciones: Mientras que en el anterior sistema, el resultado final era la media de todas las predicciones de los individuos cuyas reglas eran cumplida, en este tenemos la segunda capa, que nos produce como resultado final la media de varios subsistemas. Con ello conseguimos una especie de "opinión de varios expertos". Con ello evitamos que los resultados finales dependan en gran medida de las inicializaciones o los procesos estocásticos propios de los algoritmos. Y siempre es más fiable lo que opinen varios "expertos" que si se consultase sólo a uno. Esto queda avalado por los resultados experimentales.

- Exigencia de mínimo de puntos en la regresión: Tal y como se comentó en la sección anterior, con esto conseguimos hacer más fiables las regresiones. La contrapartida es que se presenta la posibilidad de que los sistemas desarrollados no produzcan salida para el 100 % de los patrones. En cualquier caso, al tener varios individuos (como ocurre en el SRR) o varios subsistemas (SV) lanzando predicciones, tiende a aumentar el porcentaje de datos con predicción, ya que cuando no predice un individuo o subsistema en concreto, puede que otro sí produzca predicción. La no predicción para un patrón dado ocurriría cuando ninguno diese una salida, en cuyo caso es de suponer que dicho patrón modelaba un comportamiento dificilmente predecible.

Algorithm 3 Algoritmo de Entrenamiento

variables
Set of individuals: $\{I_1, ..., I_m\}$
Set of offsprings: $\{I'_1, ..., I'_m\}$
Set of regions: $\{V_1, ..., V_m\}$
Set of regressions: $\{R_1, ..., R_m\}$
Set of training patterns: $\{T_1, ..., T_s\}$
Number of generations: $g = 0$
RandomInicialization $\{I_1, ..., I_m\}$
while $(g < GENERATIONS)$
  for (i=0) **to** m **do**
    $I'_i = Mutation(I_i)$
    $I_i = SelectBest(i, I_i, I'_i)$
  end for
end while

---

procedure SelectBest(integer i,individual J,individual J')
  $SetJ = \{I_1, ..., I_{i-1}, J, I_{i+1}, ..., I_m\}$
  $SetJ' = \{I_1, ..., I_{i-1}, J', I_{i+1}, ..., I_m\}$
  if $fitness(setJ) < fitness(setJ')$ **then** $SelectBest = J$
  else $SelectBest = J'$
end procedure

---

procedure fitness(Set of individual $\{J_1, ..., J_m\}$)
  $fitness = 0$
  CreateRegions($V_1, ..., V_m$) with $\{J_1, ..., J_m\}$
  CalculateRegression($R_1, ..., R_m$) with
    $\{V_1, ..., V_m\}$ and $\{T_1, ..., T_s\}$
  for (k=0) **to** s **do**
    find j **such** $\Pi(T_k) \in V_j$
    $fitness = fitness + |Out(T_k) - R_j(T_k)|$
  end for
end procedure

Algorithm 4 Algoritmo de predicción

PASO 1: encontrar i tal que $\Pi(T) \in V_i$
PASO 2: si ($\sharp Ri >$MIN) **entonces**
    SALIDA:= $R_i(T)$
  else SIN-SALIDA

Capítulo 6

Resultados Experimentales

Ambos sistemas han sido probados en dominios diferentes, uno artificial, la serie de Mackey-Glass y tres reales, el nivel de la marea en la laguna de Venecia, las manchas solares y el consumo de agua de un depósito. Y no sólo se ha probado la aplicación de estos algoritmos a series temporales. Y dado que los sistemas desarrollados pueden aplicarse a cualquier dominio de resgresión, también se ha apliciado al problema de predicción del rendimiento inicial de acciones (IPO). Para contrastar los resultados, se han hecho comparaciones de los mismos con los de otros algoritmos de aprendizaje automático, implementados dentro del sistema WEKA (Waikato Environment for Knowledge Analysis - Entorno para Análisis del Conocimiento de la Universidad de Waikato) [Witten et al., 1999].

Se han seleccionado 9 algoritmos capaces de producir una salida numérica, es decir, que no se usen solamente para labores de clasificación. Los algoritmos usados han sido:

- SMO-Reg: Implementa el algoritmo de optimización de secuencia mínima desarrollado por John C. Platts para entrenar una SVM (máquina de vectores soporte) con kernel polinomiales o funciones de base radial.
- IBK: está basado en el algoritmo de los k vecinos más cercanos.
- LWL: es un algoritmo genérico para el aprendizaje local con pesos. Asigna pesos usando un método basado en ejemplos y construye un clasificador a partir de los ejemplos ponderados.
- K-Star: es un clasificador basado en instancias, esto significa que la clasificación de una instancia está basada en la clasificación de instancias de entrenamiento similares, determinadas por alguna función de

similitud. Se diferencia de otros aprendizajes basados en lo mismo en que usa una función de distancia basada en entropía.

- M5Rules: genera un árbol de decisión para problemas de regresión usando una técnica de divide-y-vencerás.
- M5p: es un clasificador numérico que también combina árboles de decisión con regresiones para predecir variables continuas.
- RBNN: Red de base radial. También realiza un aprendizaje local que situa centroides para los patrones que comparten características.
- Perceptron Neural Networks: Perceptrón multicapa. Una red neuronal artificial de aproximación global para todo el conjunto de patrones.
- Conjunctive Rules: genera un conjunto de reglas para predecir tanto valores numéricos como clases nominales. La predicciones numéricas también se realizan a partir de regresiones.

Estadísticamente, a la hora de calcular una regresión, se suele requerir una cantidad mínima de puntos por cada variable para poder considerarla fiable. Dado que los sistemas presentados en esta tesis se basan en regresiones, normalmente exigiremos una mínima cantidad de puntos (a la cual nos referiremos como 'Umbral'). Si en la descripción de los problemas abordados se ha omitido la información referente al mínimo de puntos exigida, se considerará el valor por defecto como 5 puntos por cada variable de entrada, a la hora de realizar una predicción. Esta exigencia mínima nos pareció bastante razonable. En caso de que una regla o una región de Voronoi no hayan conseguido esta cantidad mínima de patrones durante la fase de entrenamiento, se considerará que no producen salida en la fase de validación.

A la hora de calcular la significancia estadístisca de los resultados obtenidos por los algoritmos desarrollados en esta tesis en comparación con los demás, se han realizado t-tests. Los valores utilizados para la constante alpha han sido 0.05 y 0.01, y los símbolos usados para dichas tablas se encuentran descritos en la Tabla 6.1.

6.1. Mareas de Venecia

Las mareas inusualmente altas son la resultante de una combinación de elementos climáticos caóticos con los sistemas de mareas periódicos asociados a un área determinada. La predicción del nivel de la marea siempre

Tabla 6.1: Valores usados en las tablas de significancia estadística

Valor	Significado
=	No significativo
+	El algoritmo de la fila es mejor que el de la columna, para un valor de alfa de 0.05
++	El algoritmo de la fila es mejor que el de la columna, para un valor de alfa de 0.01
-	El algoritmo de la fila es peor que el de la columna, para un valor de alfa de 0.05
- -	El algoritmo de la fila es peor que el de la columna, para un valor de alfa de 0.01

ha sido un tema de gran interés, y no sólo desde el punto de vista humano, sino también desde el punto de vista económico. Un buen ejemplo de esto, es el caso del nivel de la marea en la Laguna de Venecia [Michelato et al., 1983, Moretti and Tomasin, 1984]. La inundación más importante en la Laguna de Venecia se produjo en Noviembre de 1966, cuando alcanzó un nivel de cerca de 2 metros por encima de su nivel normal. Desde entonces, se han hecho muchos esfuerzos en Italia en la búsqueda de sistemas para predecir el nivel del mar en Venecia, y sobre todo, para predecir las mareas anormalmente altas [Vittori, 1992]. Durante estos últimos años, se han investigado diferentes formas de afrontar el problema, [Tomasin, 1973, Vittori, 1992], con el Perceptrón Multicapa entre otros [Zaldivar et al., 2000, Valls, 2004, Valls et al., 2006], mejorando los resultados de los modelos lineales tradicionales. Los métodos generales producen muy buenas predicciones en media, pero no tienen el mismo éxito para los valores extremos comentados.

Para este trabajo se obtuvieron los datos del nivel de la marea en la laguna de Venecia medidos cada hora a lo largo de los años 1980-1989 y 1990-1995. Estos datos fueron suministrados por A. Tomasin (Cnr-isdmg Università Ca'Foscari, Venecia). La principal peculiaridad de esta serie es que existen valores de niveles altos de mareas, que son poco corrientes pero muy importantes de predecir. Para los experimentos realizados se han usado los siguientes valores para las constantes D y τ: $D = 24$, y $\tau = 1, 4, 12, 24, 28, 48, 72, 96$. Es decir, se han usado las medidas del nivel de la marea de 24 horas consecutivas para predecir el nivel de la marea 1,4,12... etc. horas más tarde. Para los experimentos se crearon un conjunto de patrones con 24 entradas a partir de los datos de la década de los 80. Los

primeros 5000 patrones se usaron como conjunto de entrenamiento, y los 2000 siguientes como conjunto de validación. Para el sistema de reglas se usó una población de 100 individuos que evolucionaban a lo largo de 75000 generaciones, mientras que el sistema de Voronoi iteró 5000 generaciones. Esta configuración de los experimentos se muestra en la tabla 6.2.

Tabla 6.2: Configuración para los experimentos con la Serie Temporal de la Laguna de Venecia

Dominio	Laguna de Venecia
Conjunto de entrenamiento	5000
Conjunto de validación	2000
Normalización	[0,1]
Variables de entrada	24
Horizonte de predicción	1, 4, 12, 24, 28, 48, 72, 96
Medida de error	NRMSE

Los resultados de los sistemas desarrollados en esta tesis son los que aparecen en las tablas 6.3, 6.4 y 6.5. La primera tabla muestra los resultado del sistema de reglas (SRR), mientras que la segunda y tercera muentras los resultados del sistema de Voronoi (SV) para los horizontes 1,4,12 y 24 (Tabla 6.4) y 28,48,72 y 96 (Tabla 6.5). Los umbrales mencionados en estas tablas hacen referencia a los puntos mínimo exigidos a las regresiones de cada región para tomar como válida su predicción. De la tabla 6.5 se extrajeron los datos referentes al horizonte 96 para realizar los gráficos 6.1 y 6.2, en cuales se muestra la evolución del error y el porcentaje de predicción en función de los parámetros del algoritmo para dicho horizonte de preidcción.

La comparativa de los resultados obtenidos por los sistemas de aprendizaje automático implementados en WEKA, junto con los mejores resultados obtenidos por los sistemas desarrollados, pueden verse en la tabla 6.6. En dicha tabla se muestran los resultados para cada horizonte de predicción. El valor de "Porc. Pred." (Porcentaje de predicción) determina el porcentaje de puntos del fichero de validación que son predichos por al menos una regla. El resto de los puntos del conjunto de validación no pudieron ser predichos por ninguna regla del total.

Las columnas SMO-Reg, IBK, LWL, Kstar, M5Rules, M5p, Radial Basis Neural Networks, Perceptron Neural Networks and Conjuntive Rules muestran las medidas de error obtenidas por estos algoritmos. La medida de error usada en estos experimentos ha sido la medida de error estandar usada en WEKA (Raíz del Error Cuadrático Medio - Root Mean Squared Error -

RMSE), normalizada por la raíz de la varianza de la salida del conjunto de validación (NRMSE) para evitar distintas normalizaciones de la serie. Estas medidas de error se encuentran definidas en las ecuaciones 6.1 y 6.2. El gráfico 6.3 se extrajo de la Tabla 6.6, y muestra la evolución de los errores de los distintos algoritmos en función del horizonte de predicción.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} e^2}{n}} \tag{6.1}$$

$$NRMSE = \frac{RMSE}{\sqrt{Var}} \tag{6.2}$$

Tabla 6.3: NRMSE y Porcentaje de Predicción obtenidos del sistema de reglas para la Serie Temporal de la Marea de Venecia.

Pred. Hor.	Porc. pred.	NRMSE
1	98.5 %	0.0720
4	99.8 %	0.2536
12	95.2 %	0.3269
24	98.8 %	0.3547
28	98.2 %	0.4486
48	97.6 %	0.5254
72	100 %	0.6286
96	100 %	0.7057

Tabla 6.4: NRMSE y Porcentaje de Predicción obtenidos del sistema de Voronoi para la Serie Temporal de la Marea de Venecia - Primera Tabla

		Umbral	72	Umbral	120	Umbral	240
Horizonte	Regiones	NRMSE	%	NRMSE	%	NRMSE	%
1	3	0.06857	100.00	0.06857	100.00	0.06857	100.00
1	8	0.06833	100.00	0.06833	100.00	0.06841	100.00
1	12	**0.06829**	100.00	0.06833	100.00	0.06849	99.99
1	16	0.06829	99.98	0.06837	99.95	0.06833	99.60
1	20	0.06837	99.98	0.06845	99.97	0.06837	99.81
4	3	**0.24551**	100.00	0.24551	100.00	0.24552	100.00
4	8	0.25417	100.00	0.25236	100.00	0.24706	100.00
4	12	0.25693	100.00	0.25647	100.00	0.25460	100.00
4	16	0.25983	100.00	0.25815	100.00	0.25448	99.97
4	20	0.25588	99.98	0.25575	99.91	0.25086	99.54
12	3	0.42960	100.00	0.42960	100.00	0.42974	100.00
12	8	0.42755	100.00	0.41685	100.00	0.39965	99.97
12	12	0.41420	100.00	0.40496	99.97	0.38599	99.67
12	16	0.39323	99.96	0.38161	99.89	0.36591	98.94
12	20	0.40469	99.99	0.38717	99.75	**0.36155**	98.73
24	3	0.37730	100.00	0.37730	100.00	0.37873	100.00
24	8	0.37296	100.00	0.37337	100.00	0.37188	99.85
24	12	0.37126	100.00	0.37173	100.00	0.37037	99.79
24	16	0.37361	100.00	0.37233	99.90	0.36869	99.47
24	20	0.37345	100.00	0.37497	99.91	**0.36369**	99.26

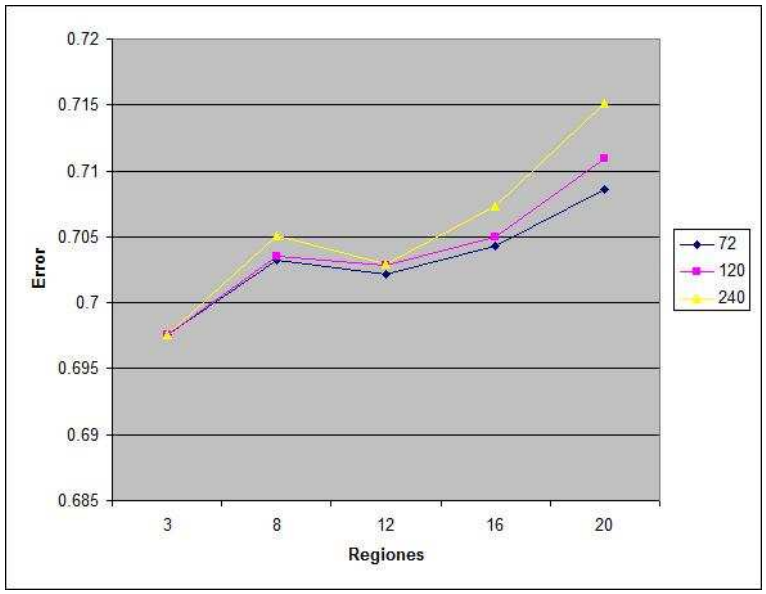

Figura 6.1:
Evolución del Error para la Serie Temporal de la Marea de Venecia

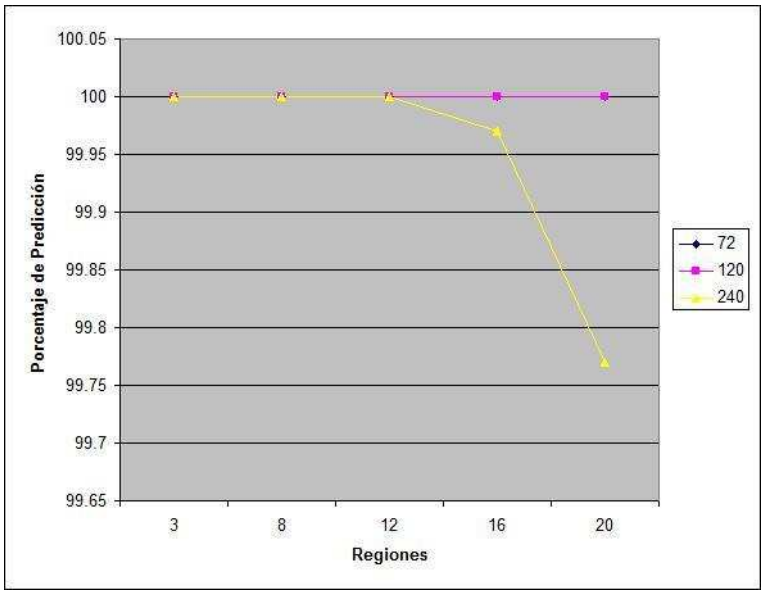

Figura 6.2:
Evolución de la predicción para la Serie Temporal de la Marea de Venecia

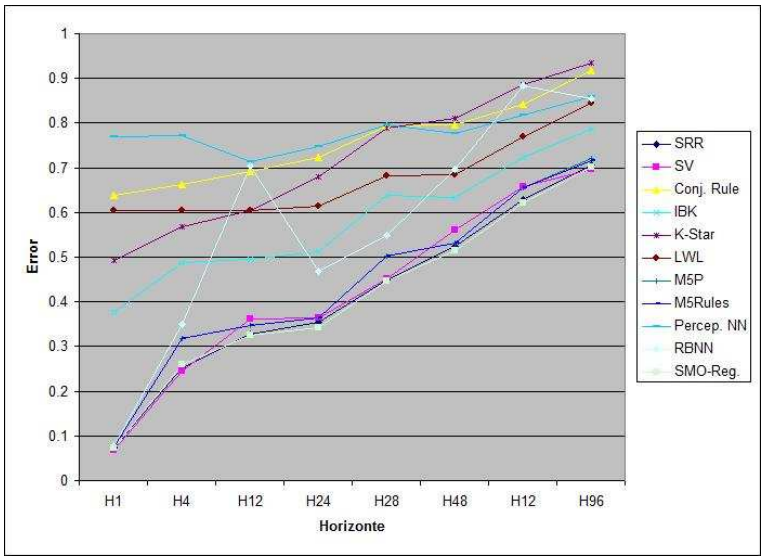

Figura 6.3:
Evolución del Error en función del Horizonte de Predicción para la Serie Temporal de las Mareas de Venecia

Tabla 6.5: NRMSE y Porcentaje de Predicción obtenidos del sistema de Voronoi para la Serie Temporal de la Marea de Venecia - Segunda Tabla

		Umbral	72	Umbral	120	Umbral	240
Horizonte	Regiones	NRMSE	%	NRMSE	%	NRMSE	%
28	3	**0.45122**	100.00	0.45163	100.00	0.45209	100.00
28	8	0.46988	100.00	0.46975	100.00	0.46886	100.00
28	12	0.46928	100.00	0.46884	100.00	0.46720	100.00
28	16	0.47130	100.00	0.47029	100.00	0.46818	100.00
28	20	0.46682	100.00	0.46722	100.00	0.46320	99.65
48	3	0.56753	100.00	0.56753	100.00	0.56753	100.00
48	8	0.56246	100.00	0.56463	100.00	0.56662	99.98
48	12	**0.56116**	100.00	0.56313	100.00	0.56353	99.99
48	16	0.56523	100.00	0.56414	100.00	0.56581	99.84
48	20	0.56353	100.00	0.56490	99.99	0.56310	99.76
72	3	0.65924	100.00	0.65924	100.00	0.65924	100.00
72	8	**0.65678**	100.00	0.65770	100.00	0.65994	100.00
72	12	0.65846	100.00	0.65904	100.00	0.65958	100.00
72	16	0.66005	100.00	0.66125	100.00	0.66224	99.84
72	20	0.65962	100.00	0.66009	99.99	0.65888	99.87
96	3	0.69759	100.00	0.69759	100.00	**0.69758**	100.00
96	8	0.70323	100.00	0.70353	100.00	0.70510	100.00
96	12	0.70215	100.00	0.70286	100.00	0.70294	100.00
96	16	0.70432	100.00	0.70500	100.00	0.70734	99.97
96	20	0.70859	100.00	0.71090	100.00	0.71514	99.77

Tabla 6.6: NRMSE y Porcentaje de Predicción obtenidos por los otros sistemas de aprendizaje automático para la Serie Temporal de la Marea de Venecia.

Algoritmo	H1	H4	H12	H24	H28	H48	H12	H96
SRR	0.0720	0.2536	0.3269	0.3547	0.4486	0.5254	0.6286	0.7057
SV	**0.0683**	**0.2455**	0.3615	0.3637	0.4512	0.5611	0.6568	**0.6976**
Conj. Rule	0.6391	0.6634	0.6926	0.7237	0.7943	0.7970	0.8420	0.9164
IBK	0.3772	0.4878	0.4948	0.5131	0.6376	0.6331	0.7221	0.7863
K-Star	0.4937	0.5688	0.6041	0.6786	0.7890	0.8098	0.8864	0.9336
LWL	0.6046	0.6036	0.6048	0.6147	0.6824	0.6842	0.7702	0.8441
M5P	0.0760	0.3174	0.3464	0.3638	0.5018	0.5309	0.6548	0.7208
M5Rules	0.0760	0.3174	0.3464	0.3638	0.5018	0.5309	0.6556	0.7159
Percep. NN	0.7698	0.7712	0.7132	0.7476	0.7964	0.7764	0.8178	0.8583
RBNN	0.0770	0.3493	0.7038	0.4679	0.5487	0.6955	0.8842	0.8537
SMO-Reg.	0.0724	0.2591	**0.3244**	**0.3432**	**0.4471**	**0.5138**	**0.6202**	0.7011

En todos los casos se ha buscado maximizar el porcentaje de datos de validación predichos sin que con ello se elevase demasiado el error medio. Para el caso del sistema de reglas, es interesante observar que cuando au-

menta el horizonte de predicción, el porcentaje de predicción no disminuye. En conclusión, el sistema parece ser estable a variaciones del horizonte de predicción. Esta propiedad resulta ser muy interesante, ya que demuestra que las reglas están adaptadas a las características especiales y locales de la serie. Para el sistema de Voronoi, se puede observar también un porcentaje de predicción del 100 % en casi todos los casos, y nunca menor de un 99.6 % .

Del estudio de la tabla 6.6 se deduce que los sistemas desarrollados en esta tesis mejoran los resultados de la mayoría de los otros algoritmos, con unos porcentajes de predicción muy cercanos al 100 %. De todo los algoritmos implementados en WEKA, SMO-Reg parece ser el único capaz de mejorar los resultados de ambos sistemas, y ni siquiera en todos los casos. Comparando los sistemas presentados en esta tesis, el sistema de Voronoi presenta unos resultados ligeramente mejores que el sistema de reglas.

En la Tabla 6.7 tenemos un estudio de la significancia estadística de los errores obtenidos en la tabla anterior para el horizonte de predicción 96. Se ha realizado la tabla únicamente para este horizonte por considerarse el de más interés. Los valores usados para la constante alfa han sido 0.05 y 0.01, y el significado de los símbolos de esta Tabla aparecen descritos el la Tabla 6.1, tal y como se indicó previamente. Como se puede observar en el estudio estadístico, los resultados del SRR son equivalentes a los del SV, salvo para el caso de 3 regiones. Dicho estudio también corrobora las conclusiones que se obtenían en la Tabla 6.6 de que los sistemas desarrollados mejoraban los resultados de los sistemas de aprendizaje automático clásicos, salvo el SMO-Reg, que obtiene unos resultados comparables, y no muy lejanos en medida de error. Es interesante, asimismo, remarcar que tanto el SRR como el SV obtuvieron un porcentaje de predicción del 100 % para sus mejores resultados en este horizonte.

En la figura 6.4 se puede ver un ejemplo de la predicción para un caso de marea inusualmente alta, con un horizonte de 1.

Tabla 6.7: Estudio estadístico de los resultados para la Serie Temporal de Mareas de Venecia.

	SRR	Rule	Conj. IBK	Kstar	LWL	M5P	M5 Rules	FF NN	RB NN	SMO Reg
SRR		++	++	++	++	++	++	++	++	- -
SV3	++	++	++	++	++	++	++	++	++	=
SV8	=	++	++	++	++	++	++	++	++	=
SV12	=	++	++	++	++	++	++	++	++	=
SV16	=	++	++	++	++	++	++	++	++	- -
SV20	=	++	++	++	++	=	=	++	++	-

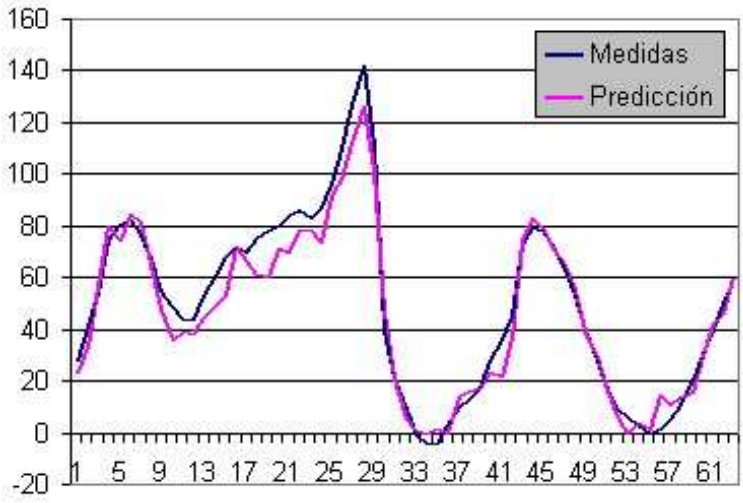

Figura 6.4:
Predicción de marea inusual con horizonte 1 por el SRR

6.2. Serie de Mackey-Glass

La serie temporal de Mackey-Glass es una serie artificial ampliamente utilizada en el campo de la predicción de series temporales, [Platt, 1991, Yingwei et al., 1997], pues tiene unas características que la hacen especialmente interesante. Se trata de una serie caótica que viene definida por la ecuación diferencial 6.3.

$$\frac{ds(t)}{dt} = -bs(t) + a\frac{s(t-\lambda)}{1+s(t-\lambda)^{10}} \tag{6.3}$$

Al igual que en los trabajos citados, se han usado los valores de las constantes $a = 0{,}2$, $b = 0{,}1$ y $\lambda = 17$. A continuación, se generaron 30000 patrones para cada horizonte de predicción. Los 4000 primeros patrones no se usarán para evitar el ruido debido a la inicialización. A continuación se usó el conjunto de datos del intervalo [5000,25000] para la labor de entrenamiento, y el segmento de datos [4000,5000] para la validación. Toda se serie se normalizó en el intervalo [0,1]. Esta configuración se resume en la tabla 6.8.

Tabla 6.8: Configuración para los experimentos con la Serie Temporal de Mackey-Glass

Dominio	Mackey-Glass
Conjunto de entrenamiento	20000
Conjunto de validación	1000
Normalización	[0,1]
Variables de entrada	24
Horizonte de predicción	50,85
Medida de error	NRMSE

Los resultados del sistema de reglas se encuentran en la tabla 6.9. En la tabla 6.10 se muestra una comparativa del error del sistema de Voronoi para el número de regiones y la cantidad de puntos mínimos exigidos a cada región. De esta última tabla se extrajeron los datos para los gráficos 6.5,6.6,6.7 y 6.8, que muestran la evolución del error y el porcentaje de predicción en función de los parámetros del algoritmo de Voronoi para los horizontes 50 y 80, respectivamente. Los resultados para estos mismos horizontes de predicción de los algoritmos IBK, LWL, KStar, M5Rules, M5p, Perceptron Neural Networks, Radial Basis Neural Networks y Conjuntive Rule implementados en WEKA, así como una regresión lineal, se muestran en la tabla 6.11, donde también se comparan los los resultados de los sistemas desarrollados en esta tesis. El error usado fue el NRMSE (normalized root mean squared error). Dicha tabla se usó para crear el gráfico 6.9, que muestra la evolución de los errores en función del horizonte de predicción. En la mayoría de los casos, el sistema de reglas mejora a la mayoría de los demás (salvo a IBK y KStar). Esto nos induce a pensar que el sistema de reglas obtiene mejores resultados en las regiones más dificiles de predecir de la Serie Temporal.

Tabla 6.9: NRMSE y Porcentaje de Predicción obtenidos por el sistema de reglas para la Serie Temporal Mackey-Glass.

Pred. Hor.	Perc. pred.	NRMSE
50	81.2 %	0.0255
85	79.6 %	0.0553

Tabla 6.10: Comparativa de los errores del sistema de Voronoi para el número de regiones de la Serie Temporal de Mackey-Glass

		Umbral	72	Umbral	120	Umbral	240
Horizonte	Regiones	NRMSE	%	NRMSE	%	NRMSE	%
50	3	0.2903	100.00	0.2903	100.00	0.2903	100.00
50	8	0.0730	100.00	0.0730	100.00	0.0735	100.00
50	12	0.0497	100.00	0.0497	100.00	0.0535	100.00
50	16	0.0342	100.00	0.0342	100.00	0.0432	98.50
50	28	0.0253	100.00	0.0260	100.00	0.0459	83.00
50	32	0.0199	100.00	0.0230	99.40	0.0453	65.30
50	40	**0.0179**	**100.00**	0.0245	99.30	0.0405	12.90
85	3	0.4365	100.00	0.4365	100.00	0.4365	100.00
85	8	0.1831	100.00	0.1831	100.00	0.1949	100.00
85	12	0.1294	100.00	0.1310	100.00	0.1535	100.00
85	16	0.0895	100.00	0.0958	100.00	0.1232	89.80
85	28	0.0676	100.00	0.0781	100.00	0.0982	77.40
85	32	0.2476	100.00	**0.0642**	**96.50**	0.1107	66.60
85	40	0.0816	100.00	0.0970	93.10	0.0862	50.30

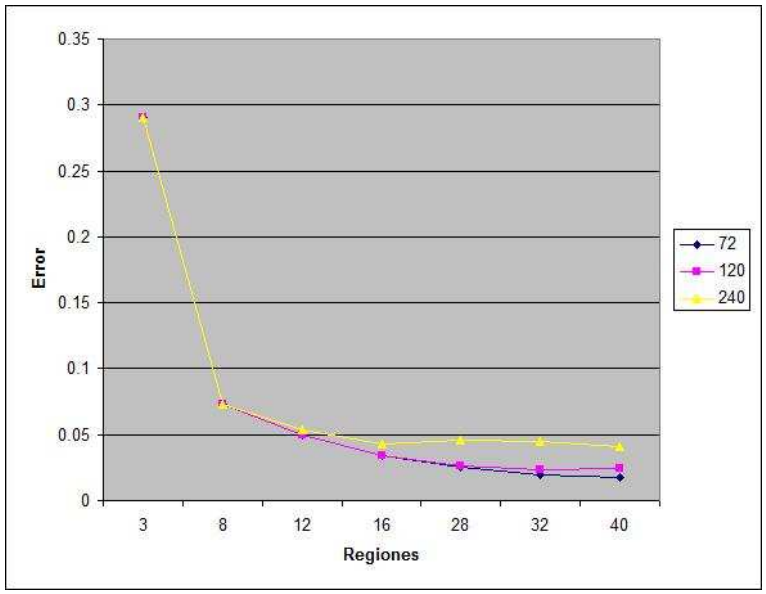

Figura 6.5:
Evolución del Error para la Serie Temporal de Mackey-Glass, Horizonte 50

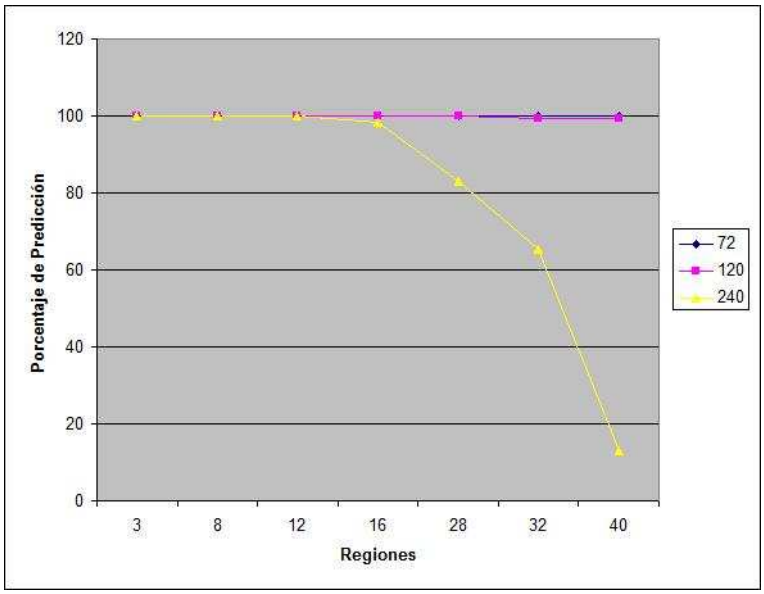

Figura 6.6:
Evolución de la predicción para la Serie Temporal de Mackey-Glass, Horizonte 50

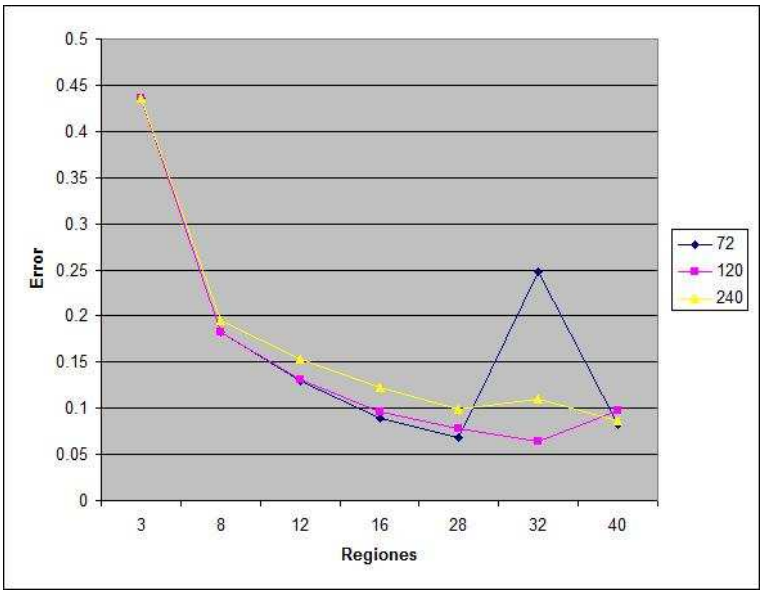

Figura 6.7:
Evolución del Error para la Serie Temporal de Mackey-Glass, Horizonte 80

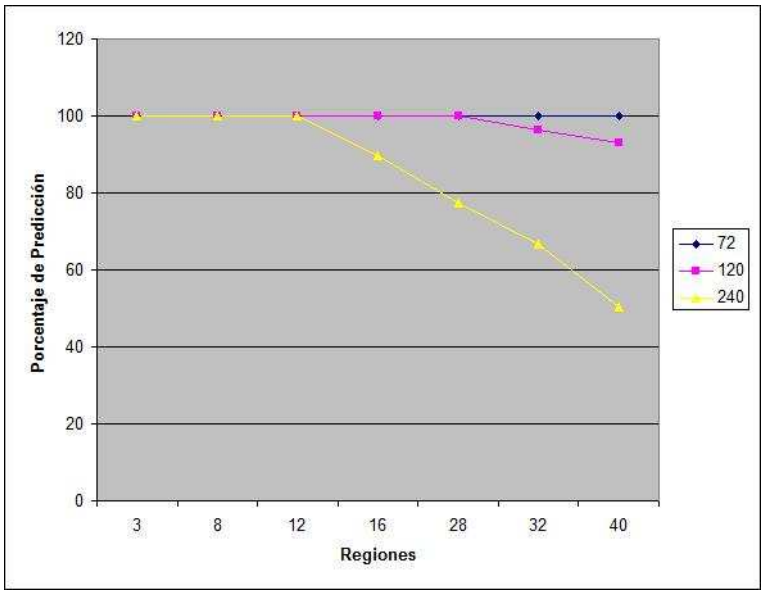

Figura 6.8:
Evolución de la predicción para la Serie Temporal de Mackey-Glass, Horizonte 80

Tabla 6.11: NRMSE obtenidos por los sistemas de aprendizaje automático para la Serie Temporal Mackey-Glass.

Algoritmo	Horizonte 50	Horizonte 80
SRR	0.0255	0.0553
SV	0.0179	0.0642
Regresión	0.0302	0.0310
Conj. Rule	0.7203	0.7827
IBK	0.0208	0.0351
K-Star	**0.0178**	**0.0286**
LWL	0.7004	0.7833
M5P	0.0503	0.0898
M5Rules	0.0660	0.1076
Percep. NN	0.1217	0.4524
RBNN	0.7045	0.7630
SMO-Reg.	0.7303	0.7507

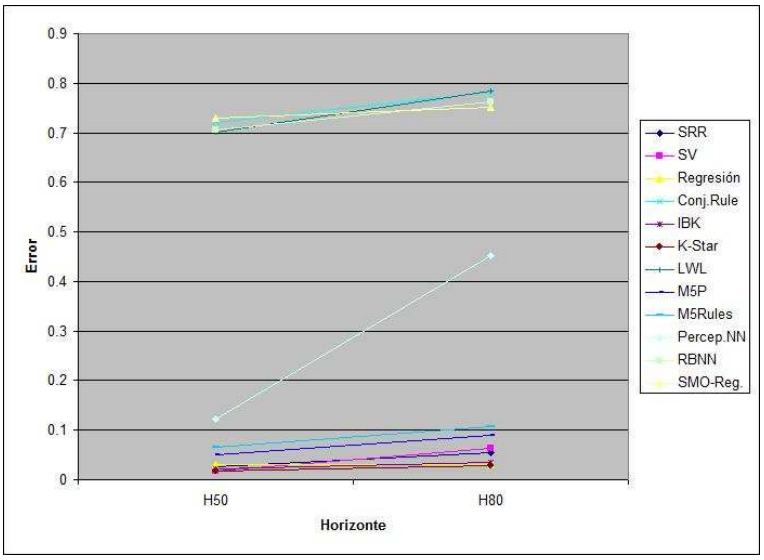

Figura 6.9:
Evolución del Error en función del Horizonte de Predicción para la Serie Temporal de Mackey-Glass

El porcentaje de predicción del Sistema de Reglas para el conjunto de validación (más del 75 %) nos induce a pensar que los elementos descartados ciertamente producían errores de predicción altos, ya que su descarte produce resultados mejores que los obtenidos por los otros sistemas.

En la Tabla 6.10 se muestran los resultados obtenidos cuando se aplica el sistema de Voronoi a los mismos conjuntos de datos. Al igual que el anterior dominio, se utilizan 3 umbrales correspondientes a 72, 120 y 240 puntos. Para los experimentos se han utilizado 3, 8, 12, 16, 28, 32 y 40 regiones. Esta tabla muestra un comportamiento uniforme para el sistema de Voronoi cuando el umbral de predicción no es demasiado alto. En este caso, el error disminuye significativamente al incrementarse el número de regiones, pero el porcentaje de predicción no cambia significativamente. Cuando incrementamos simultaneamente el número de regiones y el umbral, el sistema parece incapaz de encontrar una división del espacio de entrada con una cantidad de puntos suficientes para hacer una regresión fiable en cada región. Como se puede comprobar para otros dominios en este capítulo, un umbral entre 3 y 5 puntos por variable de entrada es más que suficiente para obtener unos resultados excelentes, siempre y cuando el número de regiones sea suficientemente alto. En la Tabla 6.11 se puede ver que los sistemas desarrollados en

esta tesis mejoran a la mayoría de los demás algoritmos, y en el caso del SV, obtiene unos resultados competitivos con los mejores de la tabla (K-Star).

Las Tablas 6.12 y 6.13 muestran un estudio de la significancia estadística de los resultados experimentales para este dominio para los horizontes 50 y 80 respectivamente. Los resultados del SV usados para esta comparativa son los obtenidos con el umbral 72 para el Horizonte 50, y umbral 120 para el Horizonte 80. Al igual que en todos los dominios presentados, los valores usados para la constante alfa han sido 0.05 y 0.01. Para el caso del Horizonte 50, se puede ver que el Sistema de reglas bate a la mayoría de los algoritmos clásicos, y este a su vez, sólo es mejorado por el sistema de Voronoi cuando tiene un gran número e regiones. En este último caso, el SV es significativamente mejor que los demás sistemas, excepto K-Star. En el caso del Horizonte 80, es el SRR el que obtiene mejores resultados que el SV. Ambos mejoran los resultados de los sistemas clásicos, salvo IBK y K-Star, que finalmente demuestran ser especialmente buenos para el caso de series temporales caóticas.

Tabla 6.12: Estudio estadístico de los resultados para la Serie Temporal de Mackey-Glass para el Horizonte 50.

	SRR	Reg.	Conj. Rule	IBK	Kstar	LWL	M5P	M5 Rules	FF NN	RB NN	SMO Reg
SRR		++	++	- -	- -	++	++	++	++	++	++
SV3	- -	++	++	- -	- -	++	- -	- -	- -	++	++
SV8	- -	++	++	- -	- -	++	- -	- -	++	++	++
SV12	- -	++	++	- -	- -	++	++	++	++	++	++
SV16	- -	++	++	- -	- -	++	++	++	++	++	++
SV28	=	++	++	- -	- -	++	++	++	++	++	++
SV32	++	++	++	++	- -	++	++	++	++	++	++
SV40	++	++	++	++	- -	++	++	++	++	++	++

6.3. Serie de manchas solares

Esta serie contiene el número medio de manchas solares medidas mensualmente desde enero de 1749 hasta marzo de 1977. Estos datos se encuentran disponibles en **http://sidc.oma.be** ("RWC Belgium World Data Center for the Sunspot"). Es una serie temporal caótica que tiene comportamientos locales, ruido en las medidas e incluso zonas que no se pueden predecir a partir del conocimiento archivado. En todos los experimentos se ha trabajado con los mismos datos: conjunto de entrenamiento desde Enero de 1749 hasta

Tabla 6.13: Estudio estadístico de los resultados para la Serie Temporal de Mackey-Glass para el Horizonte 80.

	SRR	Reg.	Conj. Rule	IBK	Kstar	LWL	M5P	M5 Rules	FF NN	RB NN	SMO Reg
SRR		++	++	- -	- -	++	++	++	++	++	++
SV3	- -	++	++	- -	- -	++	- -	- -	=	++	++
SV8	- -	++	++	- -	- -	++	- -	- -	++	++	++
SV12	- -	++	++	- -	- -	++	- -	- -	++	++	++
SV16	- -	++	++	- -	- -	++	- -	++	++	++	++
SV28	- -	++	++	- -	- -	++	++	++	++	++	++
SV32	- -	++	++	- -	- -	++	++	++	++	++	++
SV40	- -	++	++	- -	- -	++	- -	++	++	++	++

Diciembre de 1919, y validación desde Enero de 1929 hasta Marzo de 1977, normalizados en el rango $[0, 1]$; en todos los casos se usaron 24 entradas. La medida de error usada para los experimentos en este dominio ha sido la definida en la ecuación 6.4. La configuración de los experimentos se muestra en la tabla 6.14.

$$e = \frac{1}{2(N+\tau)} \sum_{i=0}^{N} (x(i) - \tilde{x}(i))^2 \tag{6.4}$$

Tabla 6.14: Configuración para los experimentos con la Serie Temporal de Manchas Solares

Dominio	Manchas Solares
Conjunto de entrenamiento	2000
Conjunto de validación	500
Normalización	[0,1]
Variables de entrada	24
Horizonte de predicción	1, 4, 8, 12, 18
Medida de error	Eq. 6.4

En la Tabla 6.15 se puede ver el error y el porcentaje de predicción para distintos horizontes para el sistema de reglas. La Tabla 6.16 muestra los errores obtenidos por el sistema de Voronoi, y por último, la Tabla 6.17 muestra los errores obtenidos por los sistemas SMO-Reg, IBK, LWL, KStar, M5Rules, M5p y Conjuntive Rule, todos ellos incluídos en WEKA. Los resultados para Perceptrón Multicapa y Redes Recurrentes se extrajeron de [Galván and Isasi, 2001]. Como en los anteriores dominios, todas estas tablas

muestran la media de 10 experimentos para todos los algoritmos con inicializaciones aleatorias. De la Tabla 6.16 se extrajeron los datos del Horizonte 18 para los Gráficos 6.10 y 6.11, que representan la dependencia del error y de porcentaje de predicción en función de los parámetros del SV. De la Tabla 6.17 se extrajeron los datos para el Gráfico 6.12, que muestra la evolución de los errores de cada algoritmo en función del Horizonte de Predicción.

Tabla 6.15: Error del sistema de reglas para la serie de Manchas Solares

Pred. Horiz.	Perc. pred.	SRR
1	100 %	0.00228
4	97.6 %	0.00351
8	95.2 %	0.00377
12	100 %	0.00642
18	99.8 %	0.01021

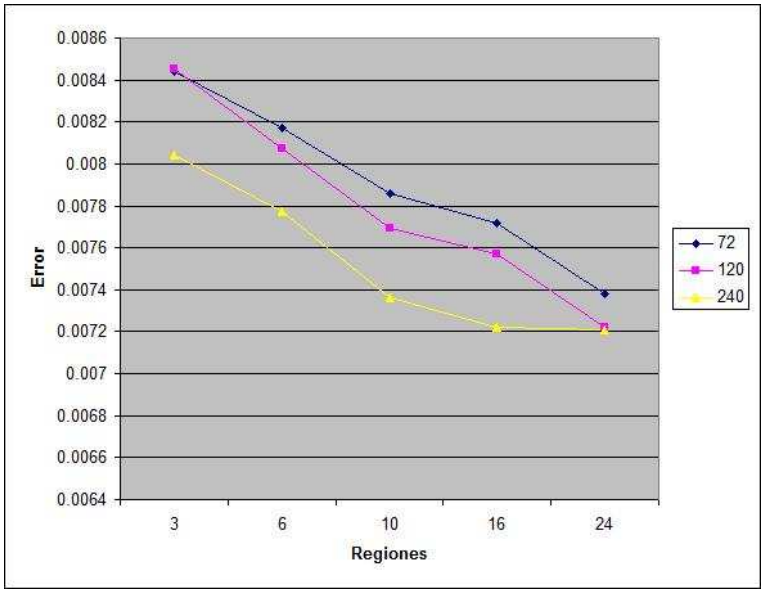

Figura 6.10:
Evolución del Error para la Serie Temporal de Manchas Solares

Tabla 6.16: Comparativa de los errores del sistema de Voronoi para el número de regiones de la Serie Temporal de Manchas Solares

Horizonte	Reg.	Umbral 72		Umbral 120		Umbral 240	
		Error	% pred.	Error	% pred.	Error	% pred.
1	3	0.00231	99.86	0.00228	99.80	0.00228	99.64
1	6	0.00224	99.52	0.00221	99.12	0.00220	98.20
1	10	0.00228	98.42	0.00213	96.54	0.00204	92.92
1	16	0.00222	96.48	0.00201	92.80	0.00183	88.28
1	24	0.00197	89.40	0.00180	86.28	**0.00165**	**81.48**
4	3	0.00332	100.00	0.00333	100.00	0.00333	99.98
4	6	0.00334	99.80	0.00335	99.74	0.00330	99.42
4	10	0.00334	99.44	0.00332	99.20	0.00317	96.20
4	16	0.00316	96.50	0.00306	95.74	0.00287	90.98
4	24	0.00289	90.74	0.00266	87.68	**0.00253**	**84.26**
8	3	0.00421	100.00	0.00399	100.00	0.00390	99.60
8	6	0.00408	99.84	0.00396	99.84	0.00391	98.88
8	10	0.00402	99.36	0.00402	98.96	0.00371	96.68
8	16	0.00384	97.86	0.00368	95.26	0.00344	90.92
8	24	0.00377	94.72	0.00354	89.80	**0.00325**	**82.24**
12	3	0.00589	100.00	0.00566	100.00	0.00533	99.30
12	6	0.00548	100.00	0.00540	100.00	0.00523	99.10
12	10	0.00541	99.78	0.00528	99.20	0.00502	96.50
12	16	0.00513	97.74	0.00503	96.10	0.00476	91.56
12	24	0.00509	92.42	0.00478	88.24	**0.00444**	**84.60**
18	3	0.00844	100.00	0.00845	100.00	0.00804	100.00
18	6	0.00817	100.00	0.00807	100.00	0.00777	99.88
18	10	0.00786	100.00	0.00769	99.76	0.00736	96.88
18	16	0.00772	98.84	0.00757	97.96	0.00722	93.12
18	24	0.00738	95.18	0.00722	93.24	**0.00721**	**86.84**

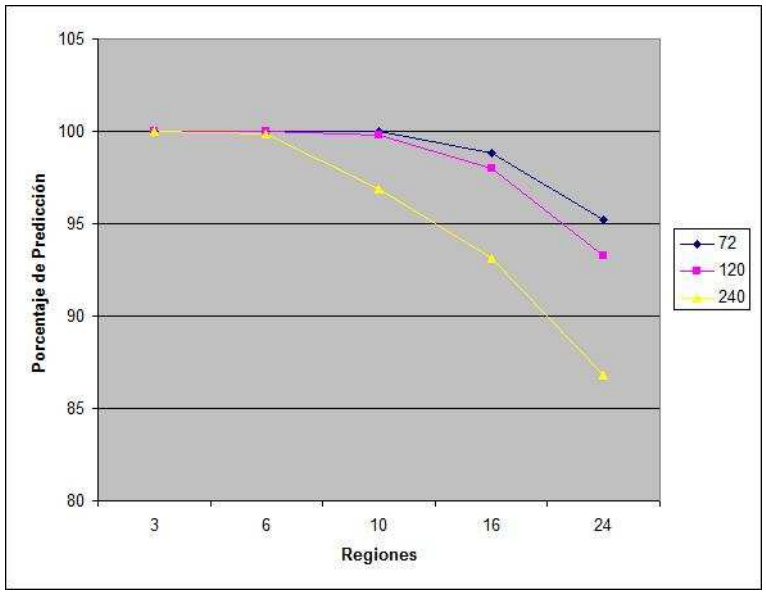

Figura 6.11:
Evolución de la predicción para la Serie Temporal de Manchas Solares

Tabla 6.17: Error para los otros sistemas de aprendizaje automático para la serie de Manchas Solares

Algoritmo	Horiz 1	Horiz 4	Horiz 8	Horiz 12	Horiz 18
SRR	0.00228	0.00351	0.00377	0.00642	0.01021
SV	**0.00165**	**0.00253**	**0.00325**	**0.00444**	**0.00721**
Regresión	0.00230	0.00384	0.00491	0.00636	0.00989
Conj. Rule	0.01211	0.01449	0.01594	0.01720	0.02040
IBK	0.00451	0.00564	0.00657	0.00794	0.01039
K-star	0.00503	0.00657	0.00868	0.01038	0.01312
LWL	0.00755	0.01004	0.01233	0.01446	0.01894
M5p	0.00230	0.00484	0.00572	0.00663	0.00807
M5Rules	0.00230	0.00397	0.00530	0.00642	0.00742
SMO-Reg	0.00233	0.00400	0.00520	0.00697	0.01131
Feedfw. NN	0.00511	0.00965	0.01177	0.01587	0.02570
Recur. NN	0.00511	0.00838	0.00781	0.01080	0.01464

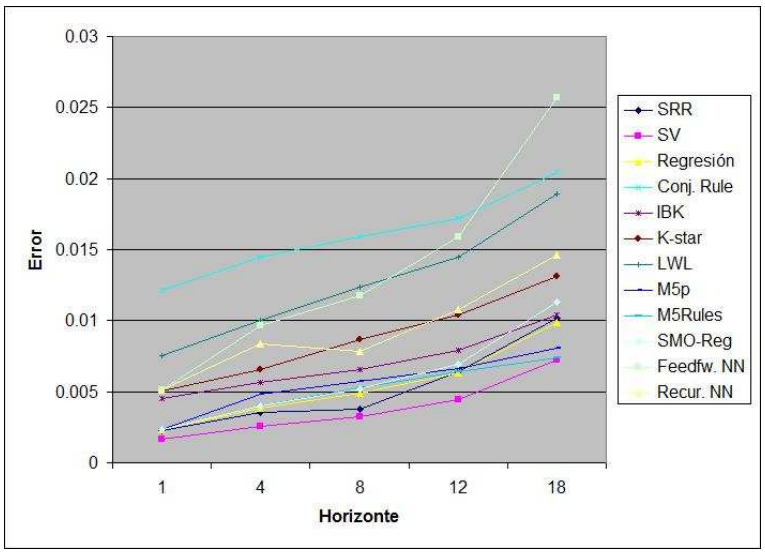

Figura 6.12:
Evolución del Error en función del Horizonte de Predicción para la Serie Temporal de Manchas Solares

Respecto al SRR, se puede observar que se obtienen mejoras frente a los resultados comparados, con un porcentaje cercano al 100 %. Si bien, la mejora obtenida por este sistema, frente a algunos de los algoritmos clásicos

es marginal para el horizonte 1, pero se va incremando con el horizonte. Un estudio más detallado de los resultados confirma la tendencia, existente en los dominios anteriores, de que este sistema es capaz de captar, de forma local, las peculiaridades de las series.

En la Tabla 6.16 comprobamos que el error se incrementa con el horizonte de predicción. Aún así, para cada horizonte el error se mantiene en unos márgenes muy bajos. Si comparamos estos resultados con los de los algoritmos clásicos (Tabla 6.17) y SRR (Tabla 6.15), vemos que los resultados son competitivos, independientemente del horizontes, cosa que no ocurría en tal medida con el SRR. Si además aceptamos trabajar con un porcentaje de predicción menor que el 100 %, los resultados pasan a ser mucho mejores que los de los demás sistemas. En definitiva, se puede concluir con que el SV obtiene un rendimiento excelente, independientemente del horizonte de predicción.

El test de significancia estadística (Tabla 6.18) se realizó sólo para el horizonte 18, al ser el caso de estudio más relevante, debido a que es el horizonte en el que los algoritmos investigados obtienen menos ventaja sobre los clásicos. Para el algoritmo SV se usó la columna correspondiente al umbral 240. Como de costumbre, los valores de la constante alfa usados fueron 0.05 y 0.01.

Tabla 6.18: Test estadístico para la serie temporal de Manchas Solares

	SRR	Reg.	Conj. Rule	IBK	Kstar	LWL	M5P	M5 Rules	SMO Reg
SRR		–	++	++	++	++	–	–	++
SV3	++	++	++	++	++	++	=	–	++
SV6	++	++	++	++	++	++	++	–	++
SV10	++	++	++	++	++	++	++	=	++
SV16	++	++	++	++	++	++	++	=	++
SV24	++	++	++	++	++	++	++	=	++

6.4. Serie del Consumo de agua

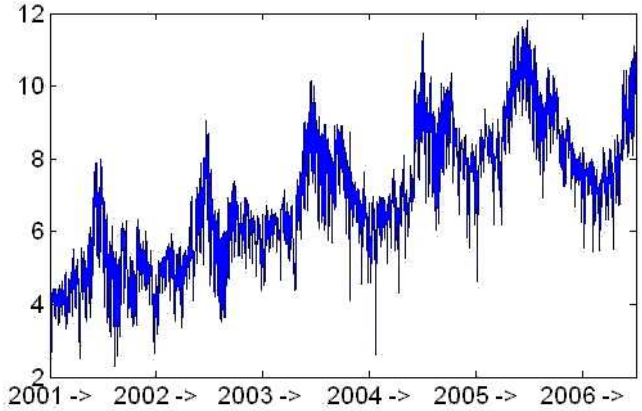

Figura 6.13:
Serie Temporal del Consumo de Agua

Esta serie temporal representa el consumo de agua de un depósito medido lo largo de 5 años y 6 meses. Se puede descargar de la página web http://atc.ugr.es/ jherrera/. Esta serie presenta comportamientos estacionarios (tal y como se puede ver en la figura 6.13), y al contrario que las anteriores que tienen una tendencia mas cíclica.

Para trabajar con la serie temporal, primero se procesaron los patrones. Para esto se normalizó la serie en el intervalo [0,1], y a continuación se crearon los patrones con 9 valores consecutivos (8 para las variables de entrada, y 1 para la variable de salida). Dado que el fichero se compone de 2007 medidas, se pudieron crear sólamente 1999 patrones, de los cuales los 1500 primeros se destinaron al conjunto de entrenamiento, y los 499 restantes al de validación. La configuración básica para estos experimentos se encuentra resumida en la Tabla 6.19.

Primero se hicieron algunos experimentos para determinar el número óptimo de regiones y de puntos mínimos para el sistema de Voronoi. Los resultados de tales experimentos se muestran en la tabla 6.20. Cada casilla de la tabla muestra la media de 10 experimentos. La columna MSE muestra el error medio, y la columna % el porcentaje medio de reconocimiento del conjunto de validación. Los Gráficos 6.14 y 6.15 están extraídos de dicha tabla. En ellos se muestra la evolución del error y el porcentaje de predicción en función de los parámetros.

Tabla 6.19: Configuración para los experimentos con la Serie Temporal del Depósito de agua

Dominio	Depósito de agua
Conjunto de entrenamiento	1500
Conjunto de validación	499
Normalización	[0,1]
Variables de entrada	8
Horizonte de predicción	1
Medida de error	MSE

Esta última tabla podemos apreciar que el error obtenido depende en gran medida del número de puntos por región (umbral de predicción), y del número de regiones. También se ve claramente que el porcentaje de predicción es bastante alto; muy cercano, de hecho al 100 %. Tal y como cabía esperar, para valores altos del umbral, se obtienen valores mas pequeños del porcentaje de predicción. Esto también ocurre cuando incrementamos el número de regiones. Para 20 regiones, dado que estamos usando 1500 patrones de entrenamiento, no podemos repartirlos entre todas las regiones asegurando que cada una recibe al menos 80 puntos: las regiones que acumulen comportamientos más similares y predecibles, agruparán mas patrones, mientras que los patrones que representan comportamientos anómalos tenderán a caer en regiones menos pobladas. Es así, pues, lógico tender a ignorar dichas regiones a la hora de predecir. Por tanto, y tal y como demuestran los resultados, esta forma de selecciónar qué patrones predecir tiende a decrementar el porcentaje de prediccion en pos de una mejora del error de predicción medio. En resumen, los resultados demuestran que el sistema de Voronoi es capaz de producir excelentes predicciones, sin depender mucho de los parámetros. También se puede comprobar que un valor de 5 por variable de entrada es un buen valor para el umbral de predicción.

Tabla 6.20: Comparativa de los errores del sistema de Voronoi para el número de regiones de la Serie Temporal del depósito de agua

Regiones	Umbral	24	Umbral	40	Umbral	80
	MSE	%	MSE	%	MSE	%
3	0.003142	100.00	0.003142	100.00	**0.003137**	**100.00**
6	0.003240	100.00	0.003229	100.00	0.003240	100.00
8	0.003346	100.00	0.003337	100.00	0.003313	100.00
10	0.003442	100.00	0.003356	100.00	0.003337	99.96
12	0.003477	100.00	0.003428	100.00	0.003413	99.94
16	0.003582	100.00	0.003443	99.70	0.003443	98.70
20	0.003803	98.96	0.003588	97.17	0.003411	92.81

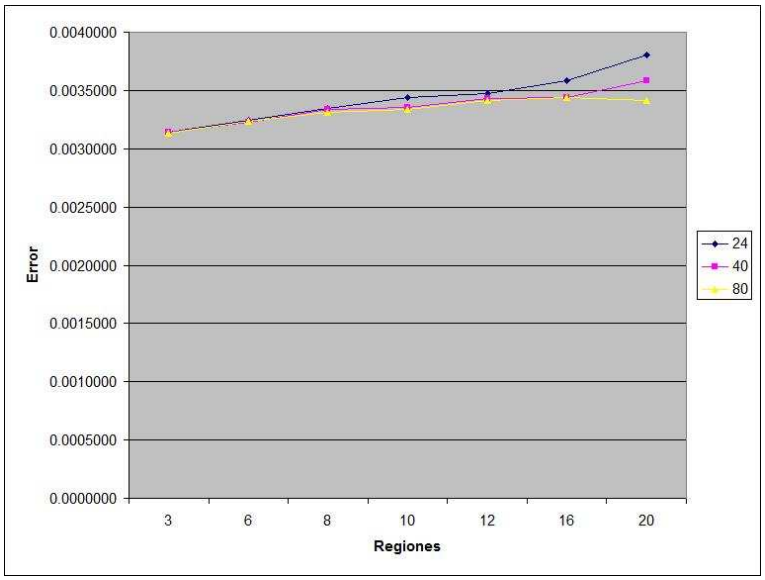

Figura 6.14:
Evolución del Error para el SV en la Serie Temporal del Consumo de Agua

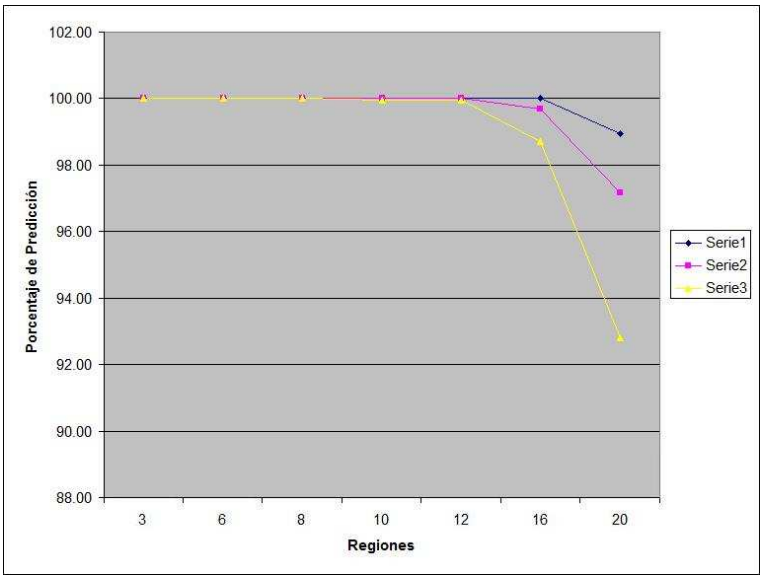

Figura 6.15:
Evolución de la predicción para el SV en la Serie Temporal del Consumo de Agua

En la tabla 6.21 se muestran los resultados obtenidos por los métodos presentados en esta tesis junto con los algoritmos seleccionados de WEKA. La fila SRR contiene el error para el Sistema de Reglas, y SV el error para el sistema de Voronoi. Para los algoritmos que requieren una inicialización aleatoria, el resultado presentado es una media de 10 experimentos. La mediada de error usada es el Error Cuadrático Medio (Mean Squared Error - MSE). El Sistema de reglas obtuvo un porcentaje de reconocimiento medio del 99,80 %.

En la Tabla 6.22 se muestra un estudio de la significancia estadística de los errores obtenidos en la tabla anterior. En esta tabla se puede observar que el Sistema de Voronoi con menos de 20 regiones es significativamente mejor que la mayoría de los sistemas de aprendizaje automático utilizados en las pruebas, para un valor de alfa de 0.01. A su vez, el Sistema de Reglas sólo es batido por el algoritmo de regiones de Voronoi, el M5P y el M5Rules, y este último resulta ser sólo ligeramente mejor.

Tabla 6.21: Comparativa de los errores para los datos del depósito de agua

Sistema	MSE
SRR	0.00382
SV	**0.00314**
Regresión	0.00379
Conj. Rule	0.02674
IBK	0.00542
Kstar	0.00629
LWL	0.02028
M5Rules	0.00377
M5P	0.00365
Perceptron	0.00673
RBNN	0.00959
SMO Reg	0.00392

Tabla 6.22: Estudio estadístico de los resultados para la Serie Temporal del depósito de agua.

	SRR	Reg.	Conj. Rule	IBK	Kstar	LWL	M5P	M5 Rules	FF NN	RB NN	SMO Reg
SRR		++	++	++	++	++	- -	-	+	++	++
SV3	++	++	++	++	++	++	++	++	+	++	++
SV6	++	++	++	++	++	++	++	++	+	++	++
SV8	++	++	++	++	++	++	++	++	+	++	++
SV10	++	++	++	++	++	++	++	++	+	++	++
SV12	++	++	++	++	++	++	++	++	+	++	++
SV16	++	++	++	++	++	++	++	++	+	++	++
SV20	++	++	++	++	++	++	+	++	+	++	++

6.5. Predicción del rendimiendo inicial de acciones (IPO)

Mucho se ha escrito sobre la existencia de extrañas tendencias en el comportamiento de los precios de las acciones el día que salen a bolsa (initial public offerings - IPOs). Esta tendencia, normalmente se corresponde con importantes ganancias, es decir, que al terminar el primer día de cotización de estas acciones, el precio de dichas acciones suele ser mucho mayor que el precio con el que se pusieron a la venta. En [Ritter and Welch, 2002] se calcula un crecimiento medio de la oferta inicial sobre el 18.8 % sobre un conjunto de 6,249 acciones de la bolsa americana entre 1980 y 2001. Este

campo de investigación lleva generando un gran interés desde hace treinta años [Ritter and Welch, 2002].

El conjunto de datos con el que se ha trabajado está compuesto por 7 variables de entrada y uno de salida. La variable de salida es rendimiento inicial de las acciones. Definiremos rendimiento inicial ajustado de una acción al cambio porcentual entre el precio de oferta al precio de cierre del primer día menos el rendimiento del índice general del mercado en que cotiza. Esto se define mediante la ecuación 6.5.

$$R_i = (\frac{Pc_i - Po_i}{Po_i}) - (\frac{Mc_i - Mo_i}{Mo_i}) \tag{6.5}$$

Donde R_i es el rendimiento inicial ajustado de la acción i, Po_i es su valor inicial de oferta, y Pc_i el precio de cierre de la acción. Así mismo, Mo_i el índice de mercado al cierre del día anterior en que la acción i fuese admitida a cotización y Mc_i el índice de mercado al cierre del día en que la acción i fue admitida a cotización.

Las variables explicativas de entrada usadas han sido:

- **Prestigio de los asesores financieros (PRESTIGIO)** Uno de los factores en el ámbito de las salidas a bolsa cuya influencia ha sido más estudiada es la composición del equipo de asesores financieros. La literatura académica ha tratado extensamente la influencia de la reputación de los asesores financieros en el comportamiento de la acción a corto plazo. Esta variable tendrá un valor binario, siendo 0 en caso de que los asesores financieros no tengan gran prestigio.

- **Rango inicial de precios (AMP-RAN)** Esta variable representa el rango orientativo de precios presentado a los inversores.

- **Revisión final del precio de venta (INDI-RAN)** La variable que propone para medir este efecto responde a la expresión:

  $$S = \frac{|P_f - P_e|}{P_e}$$

  Donde P_f es el precio final de la oferta y P_e el precio esperado definido como el punto medio del rango de oferta.

- **Precio final de oferta (PRECIO)** En [Chalk and Peavy, 1987] se sugiere que el precio debe ser incluido como variable explicativa. Los resultados de su estudio, ponen de manifiesto una correlación negativa entre el precio de emisión y el rendimiento inicial.

- **Porcentaje de capital emitido (RETE)** Esta variable es el porcentaje de capital emitido, medido como el cociente del número de acciones colocadas entre en número final de acciones que no formaron parte de la operación, será empleado para aproximar el porcentaje de capital retenido por los accionistas.

- **Tamaño de la colocación (LTAM)** Esta variable será representada a través del logaritmo del tamaño de la emisión medido en millones de dólares, excluyendo la posibilidad de sobre subscripción. La literatura académica ha documentado desde [Ritter, 1984] una relación inversa entre el tamaño de la colocación y el rendimiento.

- **Empresa tecnológica(TEC)** La razón por la que se sugiere una variable concreta para controlar si las actividades industriales de una empresa se relacionan con el sector tecnológico es el hecho de que tienden a mostrar una mayor infravaloración. Esto hecho suele ser modelizado por una variable binaria que es igual a uno para empresas tecnológicas.

Para un conocimiento más detallado de las variables de entrada, recomendamos [Quintana et al., 2005, Quintana and Isasi, 2005].

La muestra con la que se trabajó consiste en 1.036 patrones que representan las IPO de compañías que salieron al mercado entre Abril de 1996 y Noviembre de 1999 en los principales mercados estadounidenses (AMEX, NASDAQ y NYSE). La fuente de los datos fué Hoovers Online. Con idea de ordenar dichos patrones aleatoriamente, se ordenaron alfabéticamente según el nombre de la empresa. A continuación se normalizaron en el intervalo $[0, 1]$ y se enviaron los 750 primeros patrones al conjunto de entrenamiento, y los 286 restantes al de validación. El resumen de la configuración de los experimentos se encuentra en la tabla 6.23.

Tabla 6.23: Configuración para los experimentos para los datos de Bolsa.

Dominio	IPO
Conjunto de entrenamiento	750
Conjunto de validación	286
Normalización	[0,1]
Variables de entrada	7
Horizonte de predicción	1
Medida de error	NMSE

Se hicieron una serie de experimentos con el sistema de reglas para determinar el valor óptimo de la variable EMAX, así como el número de individuos. La tabla 6.24 muestra los resultados de tales experimentos, donde cada casilla muestra la media de 10 experimentos. Las columnas "EMAX" e "Individuos" muestran el valor de dichos parámetros. La columna "Reglas" muestra el número medio de las mismas, mientras que la columna NMSE muestra la media del error cuadrático medio normalizado (Normalized Mean Squared Error), y la columna % el porcentaje medio de reconocimiento del conjunto de validación. Para todos estos experimentos, a cada regla se le exigió un mínimo de 35 aciertos para ser usada.

Para el sistema de Voronoi se hicieron algunos experimentos para determinar el número óptimo de regiones y de puntos mínimos para el sistema de Voronoi. Los resultados tales experimentos se muestran en la tabla 6.25. Cada casilla de la tabla muestra la media de 10 experimentos. La columna NMSE muestra la media del error cuadrático medio normalizado, y la columna % el porcentaje medio de reconocimiento del conjunto de validación. De esta tabla se extrajeron los gráficos 6.16 y 6.17, en cuales se muestra la evolución del error y el porcentaje de predicción en función de los parámetros del algoritmo. En la tabla 6.26 se muestran los resultados obtenidos por los métodos presentados en esta tesis junto con los algoritmos seleccionados de WEKA. Vemos el algoritmo SV marcado como el mejor, seguido por el SRR.

Tabla 6.24: Comparativa de los errores del sistema de reglas para el valor de EMAX y número de individuos para el problema IPO

EMAX	Individuos	Reglas	NMSE	%
0.06	5	82.3	0.919	80.05 %
0.06	10	156.1	0.877	85.23 %
0.06	20	253.1	0.862	84.69 %
0.06	30	302.7	0.868	84.82 %
0.07	5	88.9	0.909	81.93 %
0.07	10	176.6	0.874	82.90 %
0.07	20	285.7	0.872	83.71 %
0.07	30	362.3	0.874	84.17 %
0.10	5	92.4	0.871	85.60 %
0.10	10	189.9	0.858	83.33 %
0.10	20	318.2	0.844	83.06 %
0.10	30	421.2	0.838	84.87 %
0.12	5	96.3	0.864	87.83 %
0.12	10	199.6	0.857	87.91 %
0.12	20	330.5	**0.822**	**86.04 %**
0.12	30	445.8	0.827	87.71 %
0.14	5	101.5	0.852	90.41 %
0.14	10	205.5	0.868	90.78 %
0.14	20	351.2	0.852	90.31º
0.14	30	485	0.833	90.05 %
0.16	5	108.3	0.853	91.14 %
0.16	10	223.1	0.851	92.92 %
0.16	20	371.3	0.843	93.37 %
0.16	30	521	0.839	93.20 %
0.18	10	239.9	0.855	92.39 %
0.18	20	413.6	0.843	93.45 %
0.18	20	558.1	0.855	92.50 %

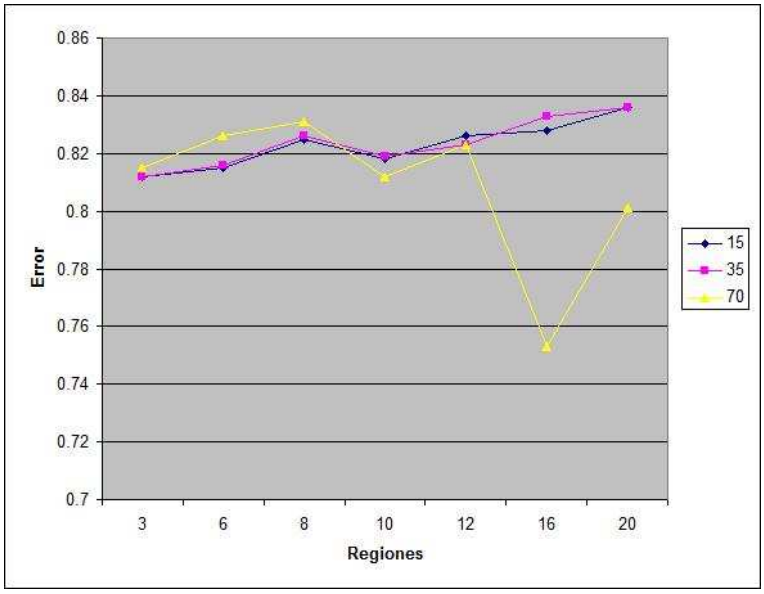

Figura 6.16:
Evolución del Error para el SV en dominio IPO

Tabla 6.25: Comparativa de los errores del sistema de Voronoi para el número de regiones del problema IPO

Regiones	Umbral NMSE	15 %	Umbral NMSE	35 %	Umbral NMSE	70 %
3	0.812	100.00	0.812	100.00	0.815	100.00
6	0.815	100.00	0.816	99.97	0.826	99.65
8	0.825	99.97	0.826	99.97	0.831	99.20
10	0.818	99.97	0.819	99.93	0.812	98.18
12	0.826	99.93	0.823	99.90	0.823	97.76
16	0.828	99.62	0.833	99.48	**0.753**	**94.30**
20	0.836	99.34	0.836	99.20	0.801	93.78

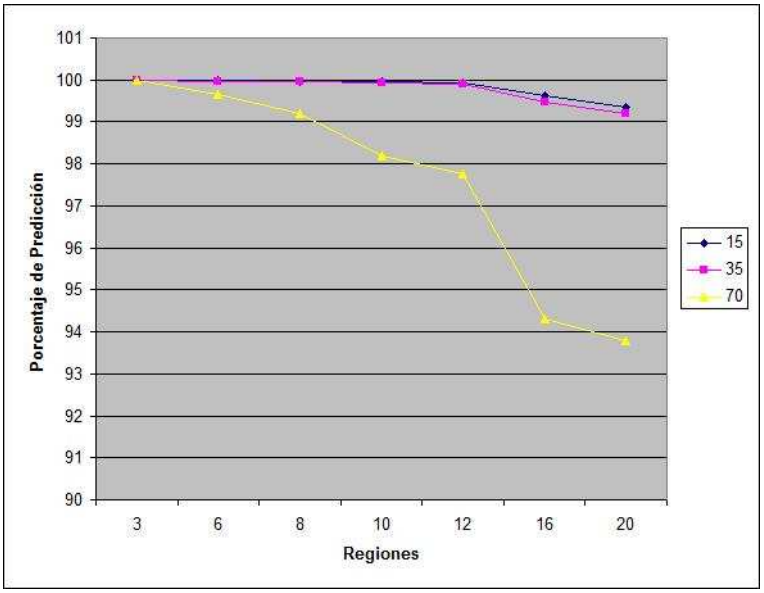

Figura 6.17:
Evolución de la predicción para el SV en dominio IPO

La Tabla 6.27 muestra los resultados del test estadístico correspondiente a este dominio. Para el SV, los datos utilizados para la comparativa se corresponden con la columna Umbral 70. Como vemos, los algoritmos presentados en este trabajo reflejan un rendimiento bastante bueno en comparación con los algoritmos de aprendizaje clásicos. Curiosamente, mientras que el caso SV16 presentaba el mejor resultado medio, el SV10 presenta un mejor

Tabla 6.26: Comparativa de los errores para los datos del problema IPO

Sistema	NMSE
SRR	0.822
SV	**0.753**
Reg.	0.890
Conj. Rule	0.976
IBK	0.837
Kstar	0.872
LWL	0.905
M5P	0.848
M5Rules	0.867
Perceptron	0.911
RBNN	0.873
SMO-Reg	0.969

resultado estadístico.

Tabla 6.27: Estudio estadístico de los resultados para los datos IPO

	SRR	Reg.	Conj. Rule	IBK	Kstar	LWL	M5P	M5 Rules	FF NN	RB NN	SMO Reg
SRR		++	++	=	++	++	+	++	++	++	++
SV3	=	++	++	++	++	++	++	++	++	++	++
SV6	=	++	++	+	++	++	++	++	++	++	++
SV8	=	++	++	=	+	++	=	+	++	+	++
SV10	=	++	++	++	++	++	++	++	++	++	++
SV12	=	++	++	=	++	++	++	++	++	++	++
SV16	=	+	++	=	+	+	=	+	++	+	++
SV20	=	++	++	+	++	++	++	++	++	++	++

6.6. Resumen de resultados

En la tabla 6.28 se muestra un resumen de los resultados obtenidos en cada dominio. En caso de que los dominios tuviesen más de un horizonte de predición, se mostrará el caso más interesante (que normalmente será el más lejano temporalmente).

Tabla 6.28: Resumen de los resultados experimentales

Dominio	Horizonte	Medida de error	1º Mejor	2º Mejor	3º Mejor
Mareas de Venecia	96	NRMSE	SV	SMO-Reg	SRR
Mackey-Glass	50	NRMSE	K-Star	SV	IBK
Mackey-Glass	85	NRMSE	K-Star	Regresión	IBK
Manchas solares	18	Eq 6.4	SV	M5Rules	M5P
Depósito de agua	-	MSE	SV	M5P	M5Rules
IPO	-	NMSE	SV	SRR	IBK

Capítulo 7

Conclusiones y líneas futuras de investigación

7.1. Conclusiones

En este tesis se han presentado dos nuevos métodos para el aprendizaje automático y su aplicación a la predicción de series temporales, basados en la búsqueda local. En general, estos algoritmos se pueden aplicar a cualquier problema de regresión. En el caso concreto de las series temporales existe una problemática proveniente de la capacidad de generalización de los sistemas de aprendizaje. Por un lado los sistemas muy generales producen buenas capacidades de predicción, sobre todo en situaciones normales de la serie, pero son incapaces de producir salidas aceptables para valores extremos. En algunos dominios esta incapacidad es crítica, ya que son los periodos anormales los que son necesarios predecir. Para solucionar este problema se han diseñado estos métodos, diseñados sobre una perspectiva Michigan.

7.1.1. Sistema de Reglas

Se ha diseñado un método basado en reglas, con perspectiva Michigan, y con selección mediante torneos y sustitución mediante estado estacionario. El método incluye un procedimiento de inicialización específico y un proceso de mantenimiento de la diversidad de las soluciones. Este método tiene la desventaja de que no siempre puede garantizar la predicción de toda la serie, pero permite alcanzar niveles de predicción muy altos a cambio de dejar una pequeña parte de la serie sin predecir.

Los resultados muestran que, sobre todo en determinadas situaciones es-

peciales (mareas alta, picos de función, etc.) el sistema es capaz de obtener resultados mejores que métodos precedentes, aunque la calidad de las predicciones en media, considerando toda la serie, no sean significativamente superiores. Es deducible, por lo tanto, que el sistema es capaz de encontrar buenas reglas para situaciones excepcionales, y no es capaz de encontrar reglas que sean mejores que las obtenidas mediante otros sistemas, para situaciones habituales, entendiendo por habituales aquellas que más se repiten en la muestra.

Otra característica importante de este sistema es que puede encontrar las regiones de la serie donde el comportamiento esté lejos de ser generalizable. Cuando la serie contiene regiones que tienen algunas particularidades especiales, el método no sólo localiza estas regiones, sino que además construye reglas particulares para predecirlas mejor. El método es también fácilmente generalizable a otros dominios diferentes de las series temporales. En general, se puede aplicar a todos los dominios de aprendizaje inductivo, siempre y cuando se posean suficientes ejemplos para el proceso de entrenamiento.

Otra característica es que siempre se pueden leer fácilmente las reglas de predicción producidas por este sistema de reglas.

7.1.2. Sistema de Voronoi

Este segundo sistema usaba Estrategias Evolutivas y una función de fitness global. De los resultados mostrados en el capítulo anterior se infiere que, definitivamente, la busqueda local mejora las aproximaciones globales.

Asimismo, este segundo sistema ha demostrado ser capaz de adaptarse mucho mejor a los datos de aprendizaje que el sistema de reglas, ya que obtiene unos porcentajes de salidas mejores que los del sistema anterior, con unas medidas de error comparables o incluso mejores.

La única característica de este sistema en la que no iguala o incluso mejora al anterior, es que el sistema predictivo no es tan fácilmente interpretable como el anterior, ya que nos devuelve una serie de puntos (que representan los prototipos de las regiones de Voronoi) y para cada uno de estos los coeficientes de la regresión asociada.

Las ventajas sobre los algoritmos de aprendizaje estándar que poseen los sistemas desarrollados en este trabajo:

- Media de diferentes predicciones: Mientras que en el sistema de Reglas, el resultado final era la media de todas las predicciones de los individuos cuyas reglas eran cumplida, en el de Voronoi tenemos la segunda capa, que nos produce como resultado final la media de varios

subsistemas. Con ello conseguimos una especie de "opinión de varios expertos". Con ello evitamos que los resultados finales dependan en gran medida de las inicializaciones o los procesos estocásticos propios de los algoritmos. Y siempre es más fiable lo que opinen varios "expertos" que si se consultase sólo a uno. Esto queda avalado por los resultados experimentales.

- Exigencia de mínimo de puntos en la regresión: con esto conseguimos hacer más fiables las regresiones. La contrapartida es que se presenta la posibilidad de que los sistemas desarrollados no produzcan salida para el 100 % de los patrones. En cualquier caso, al tener varios individuos (como ocurre en el SRR) o varios subsistemas (SV) lanzando predicciones, tiende a aumentar el porcentaje de datos con predicción, ya que cuando no predice un individuo o subsistema en concreto, puede que otro sí produzca predicción. La no predicción para un patrón dado ocurriría cuando ninguno diese una salida, en cuyo caso es de suponer que dicho patrón modelaba un comportamiento dificilmente predecible.

7.2. Líneas futuras de Investigación

Como bién queda demostrado en esta tesis, una aproximación local a las tareas de aprendizaje produce mejores resultados que una aproximación global. Y no sólo eso, ambos algoritmos también permiten detectar comprotamientos anómalos, que en la fase de entrenamiento serán ignorado, y en la de test o predicción, no producirán salida, evitando así errores mayores.

En conclusión, ambos sistemas están especialmente indicados para cualquier conjunto de datos para los cuales sea de vital importancia la detección de comportamientos atípicos, como los datos de bolsa o los fenómenos naturales que pueden inducir catástrofes.

Es importante notar que para esta tesis nos hemos centrado en la parte de la búsqueda local. A la hora de predecir, ambos sistemas admiten el uso de cualquier otro algoritmo de aproximación o aprendizaje automático en lugar de la regresión, como las redes de base radial, perceptrón multicapa, sistemas no lineales como ARMA, interpolación polinomial... etc.

También sería muy interesante determinar los valores óptimos de los parámetros de los algoritmos (EMAX, cantidad de individuos, umbral) en función de los parámetros de datos de entrada (cantidad de patrones, número de variables de entrada... etc).

Otra posible línea de investigación para el sistema de reglas sería encontrar una forma de reducir el número total de reglas producidas por sistema. Se podría hacer mediante un análisis del fichero final donde se acumulan, y posteriormente unificando las que sean muy parecidad, y aunando las que sean distintas, pero usen una regresión parecida, mediante una sintaxis que use OR u otros operadores lógicos. Esto también se podría hacer durante el proceso de entrenamiento, creando árboles lógicos mediante programación genética.

Bibliografía

[Anthony and Jennings, 2002] Anthony, P. and Jennings, N. R. (2002). Evolving bidding strategies for multiple auctions. In *ECAI*, pages 178–182.

[Axelrod, 1997] Axelrod, R. (1997). Evolving new strategies: The evolution of strategies in the iterated prisoner's dilemma. In *In Axelrod, R (Ed.): The Complexity of Cooperation : Agent-Based Models of Competition and Collaboration*, pages 10–32. Princeton University Press.

[Bäck et al., 1991] Bäck, T., Hoffmeister, F., and Schwefel, H.-P. (1991). A survey of evolution strategies. In *ICGA*, pages 2–9.

[Bäck and Schwefel, 1992] Bäck, T. and Schwefel, H.-P. (1992). Evolutionary algorithms: Some very old strategies for optimization and adaptation. In *Proc. Second Int'l Workshop Software Engineering, Artificial Intelligence, and Expert Systems for High Energy and Nuclear Physics*, pages 247–254.

[Bollerslev, 1986] Bollerslev, T. (1986). Generalized autoregressive conditional heterocedasticity. *Journal of Econometrics*, 31:307–327.

[Booker, 1982] Booker, L. B. (1982). *Intelligent behavior as an adaptation to the task environment*. PhD thesis, Ann Arbor, MI, USA.

[Booker et al., 1989] Booker, L. B., Goldberg, D. E., and Holland, J. H. (1989). Classifier systems and genetic algorithms. *Artif. Intell.*, 40(1-3):235–282.

[Box et al., 1976] Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1976). *Time Series Analysis: Forecasting and Control*. Holden-Day inc.

[Casas and Cepeda, 2008] Casas, M. and Cepeda, E. (2008). Modelos arch, garch y egarch: aplicaciones a series financieras. *Revista Cuadernos de Economía*.

[Chalk and Peavy, 1987] Chalk, A. J. and Peavy, J. W. (1987). Initial public offerings: daily returns, offering types and the price effect. *Financial Analyst Journal*, 43(5):65–69.

[Chui, 1992] Chui, C. K. (1992). *An introduction to wavelets*. Academic Press Professional, Inc., San Diego, CA, USA.

[Cleveland, 1979] Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829–836.

[Cleveland and Devlin, 1988] Cleveland, W. S. and Devlin, S. J. (1988). Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83(403):596–610.

[Dagum and Luati, 2002] Dagum, E. B. and Luati, A. (2002). Smoothing seasonally adjusted time series. In *Proceedings of the Business and Economic Statistics Section, Annual Meetings of the American Statistical Association*, pages 1347–1354.

[Engle, 1982] Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50(4):987–1007.

[Fan and Gijbels, 1996] Fan, J. and Gijbels, I. (1996). *Local Polynomial Modelling and Its Applications*. Chapman and Hall.

[Fernández and Isasi, 2004] Fernández, F. and Isasi, P. (2004). Evolutionary design of nearest prototype classifiers. *J. Heuristics*, 10(4):431–454.

[Fernández and Isasi, 2008] Fernández, F. and Isasi, P. (2008). Local feature weighting in nearest prototype classification. *IEEE Transactions on Neural Networks*, 19(1):40–53.

[Fogel, 1994] Fogel, D. B. (1994). An introduction to simulated evolutionary optimization. *IEEE transactions on neural networks*, 5(1):3–14.

[Galván and Isasi, 2001] Galván, I. M. and Isasi, P. (2001). Multi-step learning rule for recurrent neural models: An application to time series forecasting. *Neural Process. Lett.*, 13(2):115–133.

[Galván et al., 2001] Galván, I. M., Isasi, P., Aler, R., and Valls, J. M. (2001). A selective learning method to improve the generalization of multilayer feedforward neural networks. *Int. J. Neural Syst.*, 11(2):167–177.

[Gray and Thomson, 1990] Gray, A. and Thomson, P. (1990). Comments on stl: A seasonal trend decomposition procedure based on loes. *Journal of Offial Statistics*, 6:47–55.

[Gutiérrez et al., 2005] Gutiérrez, E., Taucer, F., Groeve, T. D., Al-Khudhairy, D. H. A., and Zaldivar, J. M. (2005). Analysis of worldwide earthquake mortality using multivariate demographic and seismic data. *American Journal of Epidemiology*, 161(12):1151–1158.

[Hansen and Lunde, 2006] Hansen, P. R. and Lunde, A. (2006). Consistent ranking of volatility models. *Journal of Econometrics*, 131(1-2):97–121.

[Hernández and Isasi, 2004] Hernández, J. and Isasi, P. (2004). Finding efficient distinguishers for cryptographic mappings, with an application to the block cipher tea. *Computational Intelligence*, 20:517–525.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in natural and artificial systems.* MIT Press, Cambridge, MA, USA.

[Holland and Reitman, 1977] Holland, J. H. and Reitman, J. S. (1977). Cognitive systems based on adaptive algorithms. *SIGART Bull.*, (63):49–49.

[Isasi and Hernández, 2004] Isasi, P. and Hernández, J. (2004). Introduction to the applications of evolutionary computation in computer security. *Computational Intelligence*, 20:445–449.

[Jong, 1975] Jong, K. A. D. (1975). *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, Ann Arbor, MI, USA.

[Kuzmanovski et al., 2005] Kuzmanovski, I., Trpkovska, M., and Soptrajanov, B. (2005). Optimization of supervised self-organizing maps with genetic algorithms for classification of urinary calculi. *Journal of Molecular Structure*, 744-747:833 – 838. MOLECULAR SPECTROSCOPY AND MOLECULAR STRUCTURE 2004 - A Collection of Papers Presented at the XXVIIth European Congress on Molecular Spectroscopy, Krakow, Poland, September 5-10, 2004.

[Lloyd, 1982] Lloyd, S. P. (1982). Least square quantization in pcm. *IEEE Transactions on Information Theory*, 2(28):129–137.

[Luque et al., 2004a] Luque, C., Isasi, P., and Hernández, J. (2004a). Distribución de cargas en una esfera mediante estrategias evolutivas. *Revista IEEE América Latina*, 2(2).

[Luque et al., 2004b] Luque, C., Isasi, P., and Hernández, J. C. (2004b). Forecasting time series by means of evolutionary algorithms. In *PPSN*, pages 1061–1070.

[Luque et al., 2009] Luque, C., Valls, J., and Isasi, P. (2009). Time series prediction evolving voronoi regions. *Applied Intelligence*.

[Luque et al., 2007] Luque, C., Valls, J. M., and Isasi, P. (2007). Predicción local de series temporales mediante la evolución de regiones de voronoi. In *II Simposio de Inteligenica Computacional (SICO2007)*, pages 397–402.

[Mackey and Glass, 1977] Mackey, M. and Glass, L. (1977). Oscillation and chaos in physiological control systems. *Science*, 197:287–289.

[Macqueen, 1967] Macqueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *5th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, CA.

[Meyer and Packard, 1992] Meyer, T. P. and Packard, N. H. (1992). *Local forecasting of high dimensional chaotic dynamics*.

[Michelato et al., 1983] Michelato, A., Mosetti, R., and Viezzoli, D. (1983). Statistical forecasting of strong surges and aplication to the lagoon of Venice. *Bollettino di Oceanologia Teorica ed Applicata*, 1:67–83.

[Miranda and Biles, 2007] Miranda, E. R. and Biles, J. A. (2007). *Evolutionary Computer Music*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Mitchell, 1996] Mitchell, M. (1996). *An introduction to genetic algorithms*. MIT Press, Cambridge, MA, USA.

[Mochón et al., 2005] Mochón, A., Quintana, D., Isasi, P., and Saez, Y. (2005). Genetic algorithms versus human bidding strategies for auctions. In *Fourth IEEE International Workshop on Soft Computing as transdisciplinary Science and Technology*.

[Mochón et al., 2008] Mochón, A., Quintana, D., Sáez, Y., and Isasi, P. (2008). Soft computing techniques applied to finance. *Applied Intelligence*.

[Moody and Darken, 1989] Moody, J. and Darken, C. (1989). Fast learning in networks of locally tuned processing units. *Neural Computation*, 1:281–294.

[Moretti and Tomasin, 1984] Moretti, E. and Tomasin, A. (1984). Un contributo matematico all-elaborazione previsionale dei dati di marea a Venecia. *Bollettino di Oceanologia Teorica ed Applicata*, 1:45–61.

[Nedjah et al., 2007] Nedjah, N., Abraham, A., and de Macedo Mourelle, L., editors (2007). *Computational Intelligence in Information Assurance and Security*, volume 57 of *Studies in Computational Intelligence*. Springer.

[Nelson, 1991] Nelson, B. D. (1991). Conditional heterocedasticity in asset returns: A new approach. *Econometrica*, 59(2):347–370.

[Packard, 1990] Packard, N. H. (1990). A genetic learning algorithm for the analysis of complex data. *Complex Systems*, 4(5):543–572.

[Papalexopoulos and Hesterberg, 1990] Papalexopoulos, A. and Hesterberg, T. (1990). A regression-based approach to short-term system load forecasting. *Power Systems, IEEE Transactions on*, 5(4):1535–1547.

[Park et al., 1991] Park, D., El-Sharkawi, M., Marks, R.J., I., Atlas, L., and Damborg, M. (1991). Electric load forecasting using an artificial neural network. *Power Systems, IEEE Transactions on*, 6(2):442–449.

[Pavlidis et al., 2004] Pavlidis, N. G., Tasoulis, D. K., Vrahatis, M.Ñ., and Words, K. (2004). Time series forecasting methodology for multiple-step-ahead prediction. In *The IASTED International Conference on Computational Intelligence (CI 2005*, pages 456–461.

[Platt, 1991] Platt, J. (1991). A resource-allocating network for function interpolation. *Neural Computation*, 3:213–225.

[Poon and Granger, 2003] Poon, S.-H. and Granger, C. W. J. (2003). Forecasting volatility in financial markets: A review. *Journal of Economic Literature*, 41(2):478–539.

[Pulido et al., 2005] Pulido, J. A. G., Vega-Rodríguez, M. A., Criado, J. M. G., and Sánchez-Pérez, J. M. (2005). Sunspot series prediction using adaptive identification. In *ICINCO*, pages 224–228.

[Quintana and Isasi, 2005] Quintana, D. and Isasi, P. (2005). Revisión de precios y reputación de asesores financieros: dos propuestas de índices para explicar el rendimiento a corto plazo de las salidas a bolsa. *Estudios Gerenciales*, (94):47–64.

[Quintana et al., 2005] Quintana, D., Luque, C., and Isasi, P. (2005). Evolutionary rule-based system for ipo underpricing prediction. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2005)*, pages 983– 989.

[Rahman and Hazim, 1993] Rahman, S. and Hazim, O. (1993). A generalized knowledge-based short-term load-forecasting technique. *Power Systems, IEEE Transactions on*, 8(2):508–514.

[Ritter, 1984] Ritter, J. R. (1984). The 'hot issue' market of 1980. *Journal of Business*, 57(2):215–241.

[Ritter and Welch, 2002] Ritter, J. R. and Welch, I. (2002). A review of ipo activity, pricing, and allocations. *Journal of Hydroinformatics*, 57(4):1795–1828.

[Romero and Machado, 2007] Romero, J. and Machado, P., editors (2007). *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*. Natural Computing Series. Springer Berlin Heidelberg.

[Sáez et al., 2005] Sáez, Y., Isasi, P., and Segovia, J. (2005). Análisis comparativo de la eficacia de algoritmos de iec para la generación de logotipos. In *V Congreso Españnol sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, MAEB 2005*.

[Sáez et al., 2004] Sáez, Y., Isasi, P., Segovia, J., and Hernández, J. (2004). Computación evolutiva aplicada a la generación de arte y música. In *Proceedings, Metaheurísticas y algortitmos evolutivos y bioinspirados*.

[Schwefel, 1965] Schwefel, H. P. (1965). *Kybernetische Evolution als Strategie der exprimentellen Forschung in er Strömungstechnik*. PhD thesis.

[Smith, 1980] Smith, S. F. (1980). *A learning system based on genetic adaptive algorithms*. PhD thesis, Pittsburgh, PA, USA.

[Smith, 1983] Smith, S. F. (1983). Flexible learning of problem solving heuristics through adaptive search. In *Proc. Eighth International Joint Conference on Artificial Intelligence*, pages 422–425.

[Somervuo and Kohonen, 1999] Somervuo, P. and Kohonen, T. (1999). Self-organizing maps and learning vector quantization for feature sequences. *Neural Process. Lett.*, 10(2):151–159.

[Tomasin, 1973] Tomasin, A. (1973). A computer simulation of the Adriatic Sea for the study of its dynamics and for the forecasting of floods in the town of Venice. *Comp. Phys. Comm.*, 5:51.

[Valls et al., 2006] Valls, J., Galván, I., and Isasi, P. (2006). Improving the generalization ability of rbnn using a selective strategy based on the gaussian kernel. *Computers and Informatics*, 25(1-15).

[Valls, 2004] Valls, J. M. (2004). *Selección Diferenciada del Conjunto de Entrenamiento en Redes de Neuronas mediante Aprendizaje Retardado.* PhD thesis.

[Valls et al., 2007] Valls, J. M., Aler, R., and Fernández, O. (2007). Evolving generalized euclidean distances for training rbnn. *Computers and Artificial Intelligence*, 26(1).

[Vittori, 1992] Vittori, G. (1992). On the chaotic features of tide elevation in the lagoon Venice. *Proc. of the ICCE-92, 23rd International Coference on Coastal Engineering*, pages 4–9.

[Vrahatis et al., 2002] Vrahatis, M.Ñ., Boutsinas, B., Alevizos, P., and Pavlides, G. (2002). The new k-windows algorithm for improving the k-means clustering algorithm. *Journal of Complexity*, 18(1):375 – 391.

[Wei and Billings, 2006] Wei, H. and Billings, S. (2006). An efficient nonlinear cardinal b-spline model for high tide forecasts at the venice lagoon.

[Witten et al., 1999] Witten, I., Frank, E., Trigg, L., Hall, M., Holmes, G., and Cunningham, S. (1999). Weka: Practical machine learning tools and techniques with java implementations. In *Proc ICONIP/ANZIIS/ANNES'99 Int. Workshop: Emerging Knowledge Engineering and Connectionist-Based Info. Systems.*, pages 192–196.

[Yingwei et al., 1997] Yingwei, L., Sundararajan, N., and Saratchandran, P. (1997). A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural Computation*, 9:461–478.

[Zaldivar et al., 2000] Zaldivar, J., Gutiérrez, E., Galván, I., Strozzi, F., and Tomasin, A. (2000). Forecasting high waters at Venice Lagoon using chaotic time series analysis and nonlinear neural networks. *Journal of Hydroinformatics*, 2:61–84.

[Zuo et al., 2004] Zuo, J., Tang, C., Li, C., an Yuan, C., and long Chen, A. (2004). Time series prediction based on gene expression programming. In Li, Q., Wang, G., and Feng, L., editors, *Advances in Web-Age Information Management: 5th International Conference, WAIM 2004*, volume 3129 of *Lecture Notes in Computer Science*, pages 55–64, Dalian, China. Springer.